УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютеризовані системи управління»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Система проведення турнірів з програмування»

Виконав (-ла):
студент (-ка) IV курсу, групи ІА-62
Антоненко Артем Андрійович _____

Керівник:
декан ФІОТ, доктор технічних наук, професор,
Теленик Сергій Федорович _____

Консультант:
зав. каф. АУТС, доктор технічних наук, професор,
Ролік Олександр Іванович _____

Рецензент:
доцент, кандидат технічних наук,
Волокита Артем Миколайович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Антоненку Артему Андрійовичу

1. Тема проєкту «Система проведення турнірів з програмування», керівник проєкту Теленик Сергій Федорович, доктор технічний наук, професор, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту 09.06.2020

3. Вихідні дані до проєкту : відмовостійка тестуюча система, технічні вимоги до вузла кластеру, конфігураційні файли.

4. Зміст пояснювальної записки: аналіз тестуючих систем, розробка архітектури системи проведення олімпіад, розгортання тестуючої системи, проведення турніру з програмування

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо): схема резервування фізичного рівня кластеру, схема взаємодії сервісів кластеру, архітектура тестуючої системи ejudge, діаграма можливостей користувачів системи ejudge

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ролік О.І.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Огляд та аналіз існуючих рішень	До 17.04.2020	
2	Визначення технічних характеристик	До 24.04.2020	
3	Аналіз підходів до забезпечення високої надійності	До 30.04.2020	
4	Аналіз вимог до надійності вузла кластеру	До 08.05.2020	
5	Забезпечення відмовостійкості бази даних	До 15.05.2020	
6	Забезпечення відмовостійкості файлової системи	До 19.05.2020	
7	Визначення вимог до енергоживлення	До 20.05.2020	
8	Побудова відмовостійкої тестуючої системи	До 26.05.2020	
9	Тестування розробленої системи	До 1.06.2020	

Студент Артем АНТОНЕНКО

Керівник Сергій ТЕЛЕНИК

АНОТАЦІЯ

Антоненко А.А. Система проведення турнірів з програмування. КПІ ім. Ігоря Сікорського, Київ, 2020.

Проект містить 84 сторінок тексту 48 рисунків, 1 таблицю, посилання на 26 літературні джерела, додатки.

Ключові слова: ejudge, ceph, galera, тестуюча система.

Об'єктом дослідження є міжнародна олімпіада з програмування «KPI Open».

Метою даного проєкту є підвищення ефективності проведення олімпіад, а також накопичення і документування отриманих матеріалів.

В ході виконання дипломного проєкту досліджені тестуючі системи. Розроблена та протестована відмовостійка архітектура для тестуючої системи ejudge згідно визначених умов. Розроблена система справно функціонує і вирішує поставлені перед нею завдання.

SUMMARY

Antonenko A.A. Tournament system. tournament system, Kyiv, 2020.

The project contains 84 pages of text, 48 figures, 1 table, references to 26 literature sources, appendices.

Key words: ejudge, ceph, galera, testing system.

The object of study is the international programming competition "KPI Open".

The purpose of this project is to increase the efficiency of the Olympiads, as well as the accumulation and documentation of the received materials.

During the implementation of the diploma project testing systems were studied. A fault-tolerant architecture has been developed and tested for the ejudge test system according to defined conditions. The testing system functions properly and solves the tasks set before it.

Пояснювальна записка

до дипломного проєкту

на тему: «Система проведення турнірів з програмування»

Київ - 2020

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

CDN – мережа доставки та дистрибуції вмісту.

DNS – система доменних імен.

ПЗ – програмне забезпечення.

ВСТУП

Кінець двадцятого і початок XXI століть характеризується бурхливим розвитком інформаційних технологій. Інформаційні технології все більше і більше стають невід'ємною частиною нашого життя, проникають в усі процеси життєдіяльності, допомагаючи їм розвиватися, є супутнім і одночасно невід'ємним засобом надання та аналізу інформації[1].

З розвитком комп'ютерного програмування росте кількість людей, у котрих виникає інтерес до програмування у якості спортивної дисципліни. Олімпіадне програмування вже стало невід'ємною частиною життя школярів та студентів, кожного року навчальними закладами влаштовуються змагання з програмування.

Вже давно в закладах освіти використовуються автоматизовані тестуючі системи, які здатні полегшити проведення олімпіад організаторам та учасникам. Такі системи мінімізують вплив людського фактору на проведення турнірів з програмування та прискорюють процес перевірки завдань.

Метою даного проєкту є підвищення ефективності проведення олімпіад, а також накопичення і документування отриманих матеріалів.

Для досягнення мети необхідно виконати наступні завдання:

– виконати огляд існуючих тестуючих систем;

– проаналізувати підходи до побудови відмовостійкої архітектури;

– вибрати засоби для забезпечення відмовостійкості з урахуванням вимог системи;

– забезпечити відмовостійкість тестуючої системи;

– виконати перевірку роботи розробленої системи.

Актуальність теми роботи обумовлена тим, що сучасні системи для проведення олімпіад розраховані на використання лише з використанням одного фізичного серверу. При проведенні олімпіад виникає необхідність в стабільності системи. Відмова системи або збій повинні усуватися протягом

5-15 хв, для того, щоб користувачі, без значних втрат у часі, змогли продовжити розв’язування задач. Якщо затримка у роботі керуючої системи буде більшою за 30 хв – це може поставити під сумнів подальше проведення олімпіади. Саме тому система повинна бути розроблена з урахуванням стійкості до перезавантажень, збоїв, відмови обладнання.

Об'єктом дослідження є міжнародна олімпіада з програмування «KPI Open».

Предметом – підвищення ефективності проведення міжнародної олімпіади з програмування «KPI Open».

В даній роботі представлені результати виконання поставлених вище завдань дипломного проєкту.

Дипломний проєкт складається з чотирьох розділів та додатків до них.

У першому розділі описуються теоретичні основи тестуючих систем. Проводиться порівняння методів для проведення та перевірки олімпіад. У ній наводиться визначення тестуючих систем, описуються їх можливості. Крім цього, проводиться порівняльний огляд готових рішень.

В другому розділі описуються основні моделі для побудови відмовостійкої архітектури для серверних додатків. Проводиться проектування архітектури для серверного додатку з урахуванням визначених вимог. Виконується вибір компонентів, сервісів, модулів для побудови системи.

У третьому розділі створюється відмовостійка архітектура, а саме: створення файлів конфігурації системи, створення образу системи проведення, реалізацію основних функціональних можливостей, розгортання створеної системи.

У четвертому розділі розглядається процедура проведення олімпіади на прикладі розробленої системи, наводиться інструкція для учасника та адміністратора, виконується тестування розробленої системи.

1 АНАЛІЗ ТЕСТУЮЧИХ СИСТЕМ

1.1 Визначення змагань з програмування

Олімпіади з програмування зазвичай проводиться дистанційно – за допомогою мережі Інтернет чи локально – через локальну мережу. Під час олімпіади учасники намагаються вирішити завдання згідно з наведеними специфікаціями.

Олімпіади з програмування вимагають від організаторів надання списку логічних або математичних завдань для учасників змагань (кількість яких може бути від кількох десятків до кількох тисяч), а учасники мають написати комп'ютерні програми, здатні вирішувати кожне з завдань. Переможців визначають за кількістю правильно вирішених завдань за найменший час, але також можуть бути враховані інші фактори: якість вихідних даних, час виконання, розмір програми тощо. Олімпіади можна поділити на два основні типи – міжнародні та локальні.

Регіональні олімпіади відбуваються локально в мережі кампусу, школи або дистанційно. Їх метою є відбір молоді для участі в міжнародних олімпіадах та стимулювання їх творчої роботи.

Міжнародні олімпіади проводяться за підтримки спонсорів, відбувається залучення спеціалістів з інших країн та при використанні медіа та інших ресурсів. Основними завданнями міжнародних олімпіад є:

– виявлення та розвиток обдарованої молоді;

– стимулювання студентів та школярів до творчої роботи;

– формування фахівців для дослідницької та виробничої діяльності;

– відбір студентів для участі в наукових проектах університету.

Усі завдання повинні бути унікальні та оригінальними. Для їх розробки залучаються провідні фахівці в галузях наук, що гарантує складність та унікальність задач. Головна умова щоб задачі не мали готового повного та частинного розв'язку в літературі.

Звичайною практикою в тестових та дистанційних турах є пропонування до розв'язання учасникам складних задач, що також дає студентам широкий простір для прояву ініціативи та творчого пошуку. Це дозволяє зробити процес такої науково-дослідної роботи інтерактивним та цікавим для студентів.

Суттєвою перевагою заочної олімпіади є й те, що студенти мають більше часу для творчої роботи, ознайомлення зі спеціальною літературою. Відомі випадки, коли такі задачі розв’язані під час дистанційних турів олімпіад більш швидкими шляхами.

Зазвичай при проведенні регіональних олімпіад створюється спеціальна комісія з викладачів, які після проведення олімпіади перевіряють завдання на коректність. Як правило, ці завдання можна віднести до таких категорій: теорія графів, комбінаторика, геометрія, теорія чисел, структури даних та аналіз рядків [2].

З метою підвищення об'єктивності перевірки виконаних учасниками робіт, бажано, щоб кожен член журі готував по одному завданню і він же перевіряв виконання цього завдання у всіх учасників олімпіади. Така спеціалізація членів журі сприяє, крім підвищення об'єктивності оцінки рішення, ще й скороченню часу на перевірку робіт.

Перевірка робіт здійснюється в день проведення олімпіади [3]. Критерії оцінки робіт можуть бути різними. З метою попередження можливих помилок в оцінці робіт, кращі з них, що претендують на призові місця, можуть бути перевірені повторно усіма членами журі. При таких перевірках завжди існує можливість суб'єктивного рішення та не виключається можливість підкупу членів журі. В зв'язку з цими факторами створені спеціальні тестуючі системи для перевірки рішень.

На сервері проводиться створення змагання, попередня реєстрація учасників і додавання тестових задач. Учасники повинні вирішити завдання, та відправити вихідний код програми на сервер. На сервері дані задачі компілюються, виконуються, та звіряються з еталонним виводом програми,

написаної раніше розробником задачі. Результати заносяться до бази даних та відправляються учаснику. В кінці олімпіади створюється сторінка з результатами учасників, в якій враховуються час та правильність виконання задачі. При виконанні коду вираховуються задіяні ресурси, які вказують на оптимальність рішення. Для рішень, які мають витоки пам’яті або процесорного часу, завдання може бути не зараховане. В порівнянні з класичним методом перевірки задач тестуюча система має такі переваги та недоліки [4].

Переваги:

– розроблена для автоматизації процесу перевірки. Рішення, надіслане учасником олімпіади до серверу перевіриться тестуючою системою;

– прискорюється підведення підсумків, таблиця результатів генерується у режимі реального часу;

– результати перевірок усіх вихідних кодів зберігаються у базі даних;

– тести перевіряються лише на сервері, що мінімізує вплив людського фактору на вердикт (людина може помилитися при перевірці задачі або навмисно покращити бал учасника);

– наявність зворотного зв’язку – в будь-який час олімпіади учаснику відображаються результати перевірки його задач;

– інтерактивний чат з організаторами олімпіади, в якому учасник має можливість задати питання щодо задач;

– можливість дистанційної участі.

Недоліки:

– можливість виникнення помилки системи;

– можливість вразливостей системи у програмному коді;

– необхідна попередня підготовка учасників для користування системою;

– відсутня перевірка програмного коду на коректність;

– відповідність стандартам та методам розробки ПЗ;

– задачі повинні бути розроблені з урахуванням обмежень серверного обладнання.

Використання тестуючих систем при проведенні олімпіади мінімізує вплив людського фактору, зменшує час до підготовки олімпіади, що впливає на підвищення ефективності процесу проведення олімпіади.

1.2 Огляд існуючих тестуючих систем

На даний момент існує велика кількість різних інструментів і рішень для проведення змагань з програмування: contester, pcms2, pc^2, ejudge [4]. Кожен з перерахованих вище інструментів призначений для вирішення своєї конкретної задачі.

1.2.1 Особливості тестуючої системи Contester

Contester – це система для проведення турнірів і індивідуального вирішення завдань з олімпіадного програмування. Тестуюча система має можливість перевірки рішень написаних на більшості сучасних мов програмування. Система Contester автоматично розпізнає компілятори мов програмування [5], які встановлені на робочому комп'ютері. Система має такі переваги та недоліки.

Переваги:

– автоматичний вибір компіляторів;

– обмежена підтримка Windows та Linux систем;

– можливість інтеграції з Windows Active Directory.

Недоліки:

– відсутність підтримки користувачів;

– відсутність актуальних оновлень системи;

– система підтримує компілятори лише певних та застарілих версій;

– наявність відомих проблем в системі безпеки, що дозволяє учасникам підробити результати, використовуючи відомі методи рішення нечесним шляхом;

– пропрієтарне ліцензування.

– відсутня відмовостійкість.

Дана тестуюча система не має актуальних оновлень, не підтримує нові мови програмування, обмежена у виборі операційної системи, що унеможливлює використання тестуючої системи при проведенні міжнародних олімпіад.

1.2.2 Особливості тестуючої системи PCMS2

PCMS2 – це кросплатформенна тестуюча система, яка написана на мові програмування java [6]. PCMS2 потребує складних конфігураційних файлів та скриптів запуску для створення будь-яких змагань. Понад п’ятнадцять років використовується для проведення олімпіади NEERC. Система має такі переваги та недоліки.

Переваги:

– постійні оновлення системи;

– підтримка Windows та Linux систем.

Недоліки:

– відсутність дорішування задач командами;

– відсутність можливості проведення двох турнірів в один час;

– після 2004 року нові системи не викладаються до вільного доступу;

– відсутність актуальної документації;

– відсутня відмовостійкість.

Дана система не має актуальної відкритої документації, а також відкритого коду та інсталяторів. Підтримкою тестуючої системи займаються тільки розробники, що беззаперечно ускладнює її подальшу підтримку.

1.2.3 Особливості тестуючої системи Ejudge

Система ejudge призначена для проведення турнірів та олімпіад з програмування. Основний інтерфейс системи реалізований за допомогою веб-технологій, має велику кількість параметрів, можливостей [7]. Доступ до веб-інтерфейсу забезпечується веб-сервером Apache. Утиліти командного рядка і програми з текстовим інтерфейсом дозволяють реалізувати додаткові адміністративні можливості.

Основне призначення тестуючої системи ejudge – надання користувачам зручного інтерфейсу та проведення змагань з можливістю перевірки рішень в режимі реального часу. Завдяки відкритому коду має багато модулів та велику користувацьку підтримку та документацію [4]. Має багато варіацій налаштування, що дозволяє підготувати систему до проведення турів з специфічними вимогами. Система має такі переваги та недоліки.

Переваги:

– підтримка системи розробником;

– відкритий вихідний код;

– велика кількість модулів системи;

– можливість дорішування задач командами;

– можливість проведення декількох турів одночасно;

– високий рівень безпеки системи;

– генерація протоколів олімпіади;

– інтеграція з соціальними мережами;

– актуальна документація;

– паралельна компіляція;

– можливість модифікації зовнішнього вигляду застосунку;

– можливість імпорту Polygon в турніри ejudge;

– підтримка актуальних версій компіляторів.

Недоліки:

– відсутня серверна кросплатформенність;

– відсутня відмовостійкість.

Завдяки підтримці розробників та великому колу користувачів – підтримка системи є найбільш актуальною на даний момент. Під час проведення олімпіади, учасники можуть надсилати до адміністраторів системи ejudge запит щодо неоднозначності або помилки в тексті задачі. Якщо адміністратори системи погоджуються з наявністю неоднозначності або помилки, відповідне повідомлення надсилається усім учасникам змагань.

Повідомлення про прийняті розв'язки можуть бути тимчасово припинені на певний час для збереження таємниці підсумкових результатів, про що, протягом змагань, робиться відповідне оголошення. Повідомлення про відхилені розв'язки надаються протягом змагань безперервно.

Система ejudge спроектована як гнучкий і розширюваний продукт, який допускає, як комплексне використання, так і запозичення окремих компонентів.

1.2.4 Особливості тестуючої системи PC^2

Дана тестуюча система побудована на клієнт серверній архітектурі. На сервері запускається додаток серверу, на який відправляються виконані задачі з клієнтських машин [8]. Долучитися до системи можна як адміністратор, користувач, член журі, наглядач.

На комп'ютери учасників встановлюється клієнтська програма – pcc-client, тільки за допомогою програми можна відправляти завдання на перевірку. Для проведення конкурсу за допомогою тестуючої системи PC^2 повинні виконуватися наступні вимоги [9]:

– комп'ютер з встановленою програмою має знаходитися в одній локальній мережі з сервером перевірки;

– на кожному комп’ютері учасника повинен бути встановлений та налаштований клієнт.

При ініціалізації тестуючої системи кожен модуль зчитує файли конфігурації. Для запуску на комп’ютері ініціалізується Java Virtual Machine (JVM), яка повинна мати власну структуру каталогів – включаючи власну окрему копію файлів ініціалізації PC^2 у своєму акаунті. Система має певні переваги та недоліки.

Переваги:

– актуальна документація.

Недоліки:

– обмежена підтримка мов програмування;

– необхідне встановлення та налаштування програми клієнту;

– обмежена підтримка машин з операційною системою Linux;

– система конфігурується в файлах, які доступні користувачеві, що може дозволити отримати несанкціонований доступ до серверу проведення олімпіади;

– застарілий користувацький інтерфейс;

– відсутність проведення онлайн турнірів;

– відсутня відмовостійкість.

Система не відмовостійка та складна у користуванні. У 2016 році системними адміністраторами конкурсу було вирішено розгорнути веб-надбудову для PC^2, яка не змогла обробити велику кількість запитів та зупинила турнір. Час початку змагань затягнувся на 90 хв. Команди не змогли подати задачі для перевірки, і організаційним комітетом було вирішено виконати перевірку задач вручну. Ненадійність даної системи ускладнює подальше використання тестуючої системи при проведенні олімпіад міжнародного рівня.

ВИСНОВОК ДО РОЗДІЛУ 1

Виходячи з аналізу переваг та недоліків для проведення міжнародних олімпіад, включаючи олімпіаду «KPI Open», доцільно використовувати тестуючу систему ejudge. Дана тестуюча система має ряд переваг, які вирізняють її серед інших, а саме:

– відкритий вихідний код – дана перевага дозволяє підтримувати та модифіковувати систему за необхідності, створювати та використовувати допоміжні модулі та виконувати інтеграції з іншими системами;

– високий рівень безпеки системи – дана перевага дозволяє використовувати систему при проведенні міжнародних олімпіад адже безпека передачі даних та контроль доступу контролюється тестуючою системою;

– підтримка системи розробником – завдяки постійним оновленням системи має розширений функціонал та актуальну документацію, що позитивно впливає на користувацький досвід.

Усі розглянуті тестуючі системи мають суттєвий недолік – відсутність відмовостійкості, що означає повну зупинку олімпіади при збої тестуючої системи або серверу. Даний недолік може бути усунутий завдяки розгортанні відмовостійкої архітектури. Система ejudge може бути налаштована з використанням відмовостійкої архітектури, завдяки модульній структурі.

2 РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ ПРОВЕДЕННЯ ОЛІМПІАД

2.1 Постановка завдання

Кожного року в Національному технічному університеті України «Київському політехнічному інституті імені Ігоря Сікорського» проводиться міжнародна олімпіада з програмування «KPI Open». Проведення олімпіади забезпечує навчальний заклад та створений організаційний комітет. До повноважень організаційного комітету належить:

– розробка методичних рекомендацій щодо організації та проведення олімпіади;

– контроль за проведенням олімпіади на всіх етапах;

– узгодження строків проведення олімпіади;

– затвердження завдань;

– підведення підсумків олімпіади та нагородження переможців;

– перевірка розв’язків задач учасників;

– підготовка документації (програма, листи, бланки протоколів);

– реєстрація команд.

Головним завданням комітету є вибір тестуючої програми та забезпечення безперебійної роботи системи. З перерахованих у першому розділі систем, завдяки своїм перевагам над іншими, комітетом вибрана тестуюча система ejudge. За допомогою тестуючої системи, організаційний комітет виконує такі функції:

– реєстрація користувачів у системі;

– завантаження завдань;

– перевірка вирішених завдань;

– створення протоколів змагань;

– оперативний супровід учасників;

– формування турнірної таблиці.

Для підвищення ефективності проведення олімпіади поставлено наступні завдання до розроблюваної системи:

– розглянути основні компоненти тестуючої системи;

– розглянути підходи до організації відмовостійкості додатків;

– розглянути та проаналізувати вимоги до надійності вузла;

– розробити кластерне рішення для забезпечення відмовостійкості тестуючої системи;

– створити керівництво учасника та адміністратора.

Виконання даних задач дозволить покращити користувацький досвід, дозволить використовувати тестуючу систему при проведенні олімпіад міжнародного рівня та забезпечить проведення олімпіади при відмові декількох екземплярів тестуючої системи.

2.2 Підходи до організації високої надійності

Кластер – це сукупність двох або більше серверів, об’єднаних в цілісну систему з метою забезпечення відмовостійкості сервісу та розподілення навантаження між вузлами системи [10]. Кластери можна розподілити на дві групи:

– кластери високої доступності (надійності);

– кластери розподілених обчислень.

Сервери, які працюють у кластері називають вузлами, завдяки тому що вони не несуть окремої виділеної ролі. В будь-який момент часу сервер може бути замінений на інший, при цьому, це не вплине на роботу кластеру.

В обчислювальних цілях кластери використовуються в наукових дослідженнях. Для обчислювальних кластерів істотними показниками є висока продуктивність процесора в операціях над числами з плаваючою точкою і велика потужність мережі, та менш істотними – швидкість операцій

введення-виведення, яка більшою мірою важлива для баз даних і веб-сервісів.

Обчислювальні кластери дозволяють зменшити час розрахунків, у порівнянні з одиночним комп'ютером, розбиваючи завдання на паралельно виконувані завдання, які синхронізуються за допомогою мережі.

Відмовостійкі кластери створюються для забезпечення високої доступності сервісу, що надається кластером [11]. Надмірна кількість вузлів, що входять в кластер, гарантує надання послуг в разі відмови одного або декількох серверів. Відмовостійкі кластери можна розділити на дві основні групи: високої доступності з холодним резервом та високої доступності з гарячим резервом [12].

2.2.1 Кластер високої доступності з холодним резервом

Даний метод використовується для двох серверів або більше, об'єднаних у кластер. При використанні даного підходу існує активний сервер та, зазвичай слабші за характеристиками, резервні сервери. Активний вузол обробляє запити, а пасивний чекає відмови активного. Пасивний вузол починає працювати тільки після відмови активного. Для розуміння принципу роботи такого кластеру, слід розглянути, як забезпечується переключення між серверами у випадку відмови головного вузла:

– головний сервер знаходиться в активному режимі, резервний – в пасивному – очікує відмови головного;

– головний сервер знаходиться в стані відмови, резервний – в пасивному – очікує відмови головного;

– головний сервер знаходиться в стані відмови, резервний – в активному – оброблює запити;

– головний сервер знаходиться у стані готовності, резервний – оброблює запити;

– головний сервер знаходиться в активному стані, резервний в пасивному.

Кластер, побудований за цим принципом, може забезпечити високу відмовостійкість, але в момент виключення активного вузла запити можуть не оброблятися та бути загубленими, що може призвести до втрати важливих даних. Основним недоліком даного підходу є те, що переключення від головного до резервного вузла може потребувати велику кількість часу.

Даний підхід не рекомендується використовувати, коли необхідна безперервна робота системи, бо переключення між вузлами потребує велику кількість часу.

2.2.2 Кластер високої доступності з гарячим резервом

При використанні даного методу усі вузли несуть робоче навантаження, оброблюють запити. При гарячому резервуванні один сервер працює в режимі «робота», а інший – в режимі «резерв». Сервер в режимі «резерв» не відправляє контролюючі команди, а також має ряд обмежень в аспекті контролю кластеру.

При використанні даного методу, як правило, вузли мають спільну файлову систему. Сучасні розподілені файлові системи потребують використання трьох вузлів для забезпечення відмовостійкості системи. Для розуміння принципу роботи такого кластеру, слід розглянути, , як забезпечується переключення у випадку відмови вузла на прикладі трьохвузлового кластеру:

– усі вузли знаходяться в активному стані. Балансувальник навантажень направляє трафік до вузлів кластеру;

– головний вузол знаходиться в стані відмови. Балансувальник направляє трафік до інших вузлів;

– головний вузол знаходиться в активному стані, відбувається синхронізація даних між вузлами. Балансувальник направляє трафік до інших вузлів;

– головний вузол знаходиться в активному стані. Балансувальник навантажень направляє трафік до вузлів кластеру.

Кластер, побудований за цим принципом, може забезпечити високу відмовостійкість завдяки тому, що дані між вузлами синхронізуються та переключення між активним/резервним вузлом виконується автоматично зовнішнім балансувальником. Момент переключення між вузлами може зайняти декілька секунд, що дозволить не втратити важливі дані та забезпечити доступність системи при втраті вузла. Завдяки даним властивостям даний підхід доцільно використовувати при необхідності забезпечення безперебійної роботи системи.

2.3 Технічна характеристика

Кластер з гарячим резервом забезпечує високу відмовостійкість завдяки синхронізації та швидкому переключенні між вузлами у разі їх відмови. Для використання даного підходу кластер повинен складатися мінімум з трьох вузлів та мати високу пропускну здатність мережі. Вузли кластеру повинні відповідати наступним характеристикам , що наведені в таблиці 2.1.

Таблиця 2.1 – Технічні характеристики вузла

Елемент	Кількість
HDD – 250Gb;	2
RAM 16Gb	1
Intel Xeon Bronze 3104	1
Ethernet 1Gb/сек	2

2.4 Вимоги до проектування додатків

Не кожен додаток може працювати в середовищі кластера високої доступності. Для того, щоб запустити в середовищі кластера високої доступності, додаток повинен задовольняти таким технічним вимогам:

– проста процедура запуску, зупинки, перезавантаження;

– підтримка декількох екземплярів;

– можливість використовувати спільне сховище (SAN, NAS);

– додаток не повинен псувати дані при перезапуску або несправності;

– можливість відновлення додатку на іншому вузлі кластеру;

– можливість обробляти дані незалежно від доступності модулів архітектури.

Спроектований додаток повинен забезпечувати роботу, незалежно від інших додатків системи, мати можливість відновитися на іншому вузлі кластеру, при цьому, не спричинити збій у системі зберігання даних.

2.5 Вимоги до надійності вузла

Кластери високої надійності використовують дані методи для забезпечення надійності інфраструктури, на якій планується запуск додатку:

– дзеркальне відображення до диску (RAID) – відмова внутрішніх дисків серверу не призводить до збоїв системи;

– розподілене сховище даних;

– резервні мережеві з’єднання – відключення комутатору, мережевого інтерфейсу не призводить до втрати працездатності системи;

– надмірність бази даних та постійна реплікація даних;

– забезпечення надлишковості живлення – використання безперебійних джерел енергоживлення, резервування джерел живлення.

Дані методи допомагають звести до мінімуму ймовірність відмови вузла, а також дозволяють відновити систему з мінімальними втратами у наданні послуг.

2.5.1 Розподілене сховище даних

Для створення системи високої доступності необхідні додаткові підсистеми, окрім самого програмного забезпечення серверу. Сучасні додатки потребують стійкої, орієнтованої на транзакції, файлової системи. Файлова система повинна відрізнятися незначним часом перевірки цілісності та відновлення на випадок відмови вузла кластера при передачі ресурсів іншому вузлу та відмовостійкій базі даних.

Традиційне розподілене сховище даних дозволяє безлічі користувачів, що працюють з однією системою, розділяти доступ до файлів, що зберігаються локально на машині. Розподілена файлова система розширює ці можливості, дозволяючи розділяти доступ до файлів користувачам на машинах, об'єднаних між собою за допомогою мережі.

В основі розподілених файлових систем лежить модель клієнт-сервер. Під клієнтом розуміється машина, яка звертається до деякого файлу, а під сервером – машина, що зберігає файли і забезпечує до них доступ.

Розподілена файлова система являє собою певний шар програмного забезпечення, який управляє зв'язком між традиційними операційними системами і файловими системами. Цей шар програмних засобів інтегрується з операційними системами хостів мережі і забезпечує сервіс розподіленого доступу до файлів для систем, які мають централізоване ядро. Розподілені файлові системи мають такі властивості:

– мережева прозорість – клієнти повинні мати можливість звертатися до виділених файлів користуючись тими ж самими операціями, що і для доступу до локальних файлів;

– прозорість розміщення – ім’я файлу не повинно визначати його місцеположення в мережі;

– незалежність розміщення – ім’я файлу не повинно змінюватися при зміні його фізичного розплодження;

– мобільність користувача – користувачі повинні мати можливість звертатися до файлів, з будь-якого вузла мережі;

– стійкість до збоїв – система повинна продовжувати функціонувати при несправності окремого компонента (сервера або сегмента мережі);

– масштабованість – система повинна мати можливість масштабування в разі збільшення навантаження;

– мобільність файлів – повинна бути можливість переміщення файлів з одного місця розташування в інше на працюючій системі.

Для забезпечення високої відмовостійкості слід використовувати розподілене сховище для важливих файлів системи. На даний момент найпопулярнішими системами є: Ceph, HDFS, DRBD [13].

2.5.1.1 Особливості розподіленого сховища даних Ceph

Ceph є закінченою системою зберігання даних корпоративного рівня, що забезпечує підтримку для широкого діапазону протоколів і методів доступу.

Блокові системи зберігання є класом сховищ, що використовуються для зберігання даних в мережі зберігання даних. Дані зберігаються у вигляді томів, які знаходяться в формі блоків і приєднуються до вузлів. Це забезпечує більшу ємність зберігання, необхідну додатками з високим ступенем надійності і продуктивності.

Ceph використовує власний протокол RBD, який забезпечує надійні, розподілені і високо продуктивні диски зберігання блоків для клієнтів. RBD блоки чергуються з численними об'єктами, які розділені між вузлами

кластеру, тим самим забезпечуючи клієнтам надійність зберігання даних і продуктивність. RBD має вбудовану підтримку ядра Linux та забезпечує такі функції рівня підприємства:

– повні та інкрементальні моментальні знімки;

– слабку ініціалізацію ;

– клонування копії під час запису;

– кешування в пам'яті;

Ceph RBD підтримує образи розміром до 16 екзабайт. Ці образи можуть бути відображені як диски в машинах Bare Metal, віртуальних машинах або в звичайних машинах хостів. Провідні в галузі гіпервізори з відкритим кодом, такі як KVM і Zen забезпечують повну підтримку RBD і використовують можливості RBD для своїх гостьових віртуальних машин.

Файлова система Ceph має назву CephFS, є POSIX-сумісною файловою системою та має підтримку власного драйвера ядра Linux, що робить CephFS високо адаптивною для будь-якої операційної системи Linux. CephFS зберігає дані і метадані окремо, забезпечуючи тим самим збільшення продуктивності і надійності додатків. Ceph має такий ряд переваг:

– єдина, відкрита і уніфікована платформа: блок, об'єкт і зберігання файлів, об'єднані в одну платформу;

– тонке забезпечення: виділення місця є лише віртуальним, а фактичний простір на диску надається у міру необхідності. Це забезпечує більшу гнучкість та ефективність;

– реплікація: у Ceph Storage всі дані, що зберігаються, автоматично реплікуються з одного вузла на інші. Повторний примірник даних є на вузлах кластері в будь-який момент;

– відновлення: кластер постійно контролюється. Якщо одна із триразових копій відсутня, копія створюється автоматично, щоб гарантувати наявність трьох примірників;

– висока доступність: У Ceph Storage всі дані, що зберігаються, автоматично реплікуються з одного вузла на кілька інших вузлів;

– масштабованість: Ceph працює в кластерах, які можна збільшувати при необхідності, таким чином задовольняючи майбутні потреби масштабу.

Для розуміння відмовостійкості даного сховища слід розглянути відмову вузла на прикладі трьохвузлової системи (рисунок 2.1). При записі файлів до файлової системи, ceph дублює блоки файлу до вузлів системи:

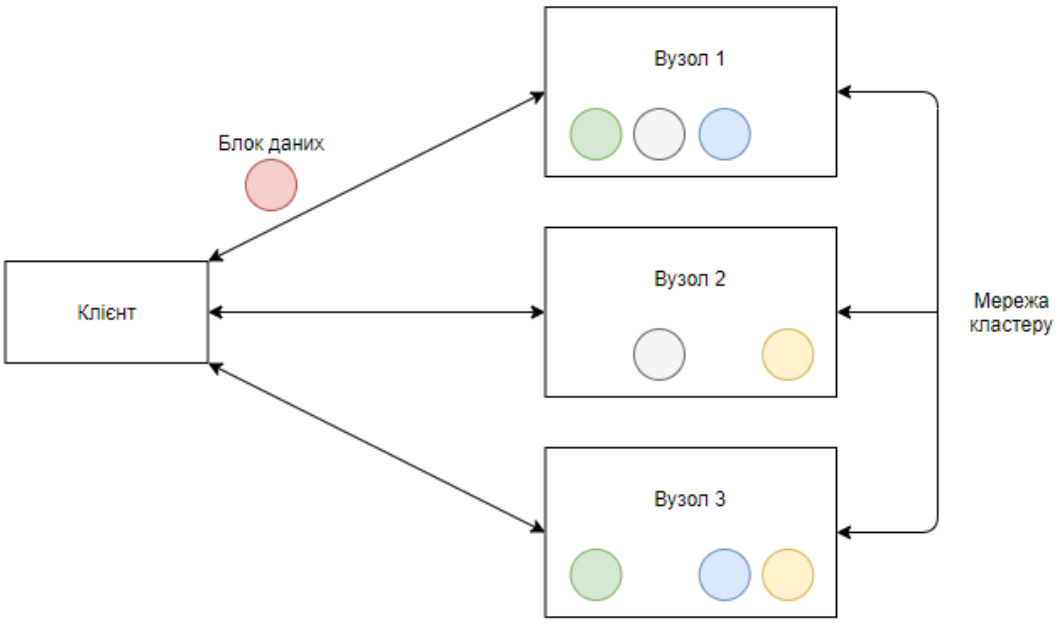


Рисунок 2.1 – Дублювання блоків даних

При виході будь-якого вузла або групи вузлів з ладу Ceph відновить втрачені копії на інших вузлах доти, поки вузли або диски, які вийшли з ладу, не замінять на робочі [14]. При цьому перебудова відбувається без простою і прозоро для клієнтів.

Після синхронізації кластеру, копії блоків файлів набудуть свою надмірність та відновлять свою високу доступність (рисунок 2.2). Для забезпечення даних функцій в Ceph передбачено використання таких головних компонентів: OSD, MON, MDS.

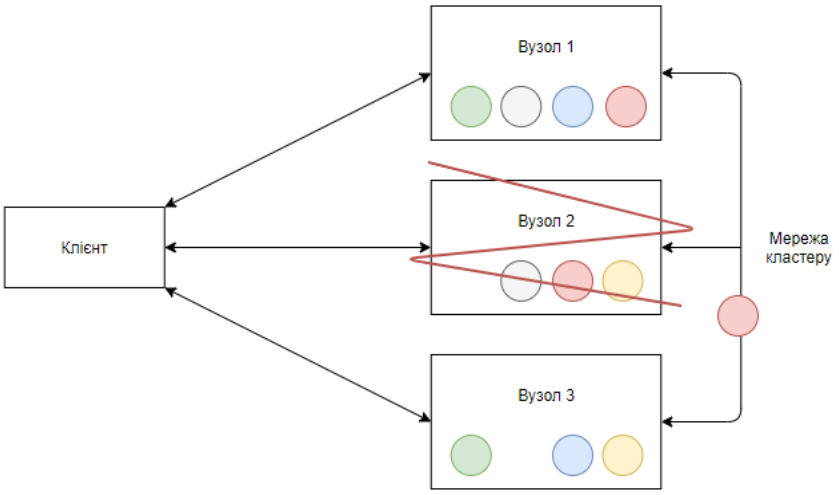


Рисунок 2.2 – Перебудова блоків даних

MON (Monitor) – це компонент, який виконує роль координатора кластеру, зберігає інформацію про стан, обмінюючись з іншими координаторами метаданими. Клієнти звертаються до MON, щоб дізнатися, на які OSD писати/читати дані. При розгортанні нового сховища, насамперед створюється MON. Кластер може працювати з використанням лише одного MON, але рекомендується використовувати три або п’ять MON для забезпечення високої відмовостійкості. Кількість MON повинна бути непарною для уникнення ситуацій роздвоєння свідомості (split-brain).

OSD (Object Storage Device) – це одиниця сховища, яка зберігає самі дані і обробляє запити клієнтів, обмінюючись даними з іншими OSD. Один MON і один OSD – мінімальний набір для того, щоб працювати з кластером. Для кожного диску та вузлу повинен створюватися окремий OSD. Клієнти працюють безпосередньо з самими OSD, минаючи вузькі місця, і досягаючи, тим самим, розподілу навантаження.

Диск OSD складається з двох частин: журнал і самі дані. Відповідно, дані спочатку записуються в журнал, а згодом, в розділ даних. З одного боку це дає додаткову надійність і деяку оптимізацію, а з іншого боку – додаткову операцію, яка може позначитися на продуктивності.

MDS (Metadata Server Daemon) – сервер метаданих, який потрібен для роботи файлової системи CephFS. Поділ метаданих від даних значно збільшує продуктивність кластера. Наприклад, для лістингу директорії немає необхідності визивати OSD, дані беруться з MDS.

Завдяки своїй архітектурі Ceph забезпечує високу швидкість роботи з файлами різного розміру та забезпечує високу відмовостійкість завдяки відсутності єдиної точки відмови.

2.5.1.2 Особливості розподіленого сховища даних HDFS

Файлова система HDFS розроблена з використанням розподіленого дизайну файлової системи. HDFS відрізняється високою відмовостійкісттю та може бути розгорнуто. з використанням апаратної частини низької вартості.

HDFS створений для зберігання великих об'ємів даних [15], які розподіляються між вузлами кластеру. На вузлах кластеру зберігаються копії файлів для їх відновлення у разі втрати одного з вузлів.

HDFS має архітектуру master/slave. Кластер складається з одного вузла контролю, який керує простором імен файлової системи та регулює доступ клієнтів до файлів. В кластері існують вузли даних, які управляють сховищем, приєднаним до вузлів, на яких вони працюють (рисунок 2.3).

HDFS відкриває простір імен файлової системи та дозволяє зберігати дані користувачів у файлах. В системі файл розділений на один або кілька блоків, які зберігаються у наборі вузлів даних. Вузол контролю виконує такі операції в просторі імен файлової системи, як відкриття, закриття та перейменування файлів і каталогів.

HDFS побудований з використанням мови Java, що забезпечує кросплатформенність на надає можливість запускати необхідні компоненти системи на операційних системах з підтримкою Java. Система складається з таких компонентів:

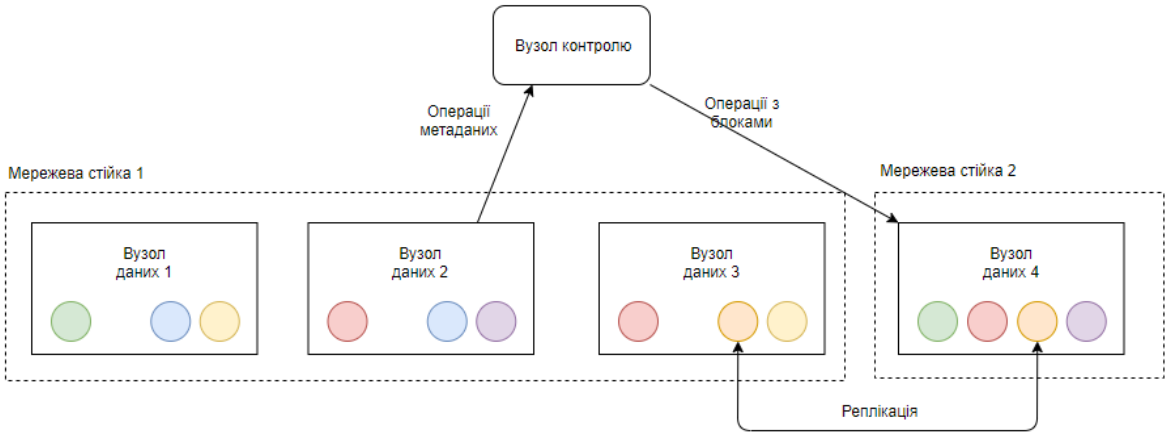


Рисунок 2.3 – Архітектура HDFS

Вузол контролю (Namenode) – є найуразливішим місцем системи та потребує найбільшої кількості виділених ресурсів. Даний компонент:

– керує простором імен файлової системи;

– регулює доступом клієнта до файлової системи;

– виконує операції з перейменування, відкриття, закриття та роботи з директоріями.

Вузол даних (Datanode) – використовується для зберігання блоків даних, повинен бути встановлений на кожному вузлі кластеру. Даний компонент виконує такі функції:

– операції запису/читання;

– створення та видалення блоку;

– реплікація файлів та блоків.

Блок – мінімальний обсяг даних, який HDFS може прочитати чи записати. Файл у файловій системі буде розділений на один або кілька сегментів та/або збережений в окремих вузлах даних. Ці сегменти файлів називаються блоками. За замовчуванням розмір блоку – 64 Мб, але його можна збільшити відповідно до необхідності зміни конфігурації HDFS

Завдяки даній архітектурі HDFS забезпечує високу відмовостійкість. При втраті третини вузлів, кластер може продовжити свою роботу, при цьому не втративши важливих даних. HDFS має такий ряд переваг:

– швидка робота з великими файлами;

– розділення файлів на блоки та подальша реплікація та збереження на вузлах кластеру;

– підтримка традиційної ієрархічної структури;

– кросплатформенність;

– можливість роботи з файловою системою за допомогою REST API.

HDFS розроблений для роботи з великими масивами файлів але при цьому відображає низьку швидкість роботи з маленькими файлами. Має єдину точку відмови у компоненті вузлу контролю.

2.5.1.3 Особливості розподіленого сховища даних DRBD

Основна функціональність DRBD реалізована за допомогою модуля ядра Linux. Зокрема, DRBD є драйвером для пристрою віртуального блоку, тому DRBD розташований біля нижньої частини стека вводу/виводу системи.

Через це DRBD є надзвичайно гнучким та універсальним, що робить DRBD рішенням для реплікації, придатним для впровадження високої доступності майже до будь-якого додатку.

На рисунку 2.4 представлений огляд DRBD в контексті двох незалежних серверів, які надають незалежні ресурси зберігання. Один із серверів визначається як основний, а другий – вторинний.

Користувачі отримують доступ до блокових пристроїв DRBD як до традиційних локальних пристроїв блоку або мережевих рішень для зберігання даних. Програмне забезпечення DRBD забезпечує синхронізацію між первинним та вторинним серверами для операцій читання та запису.

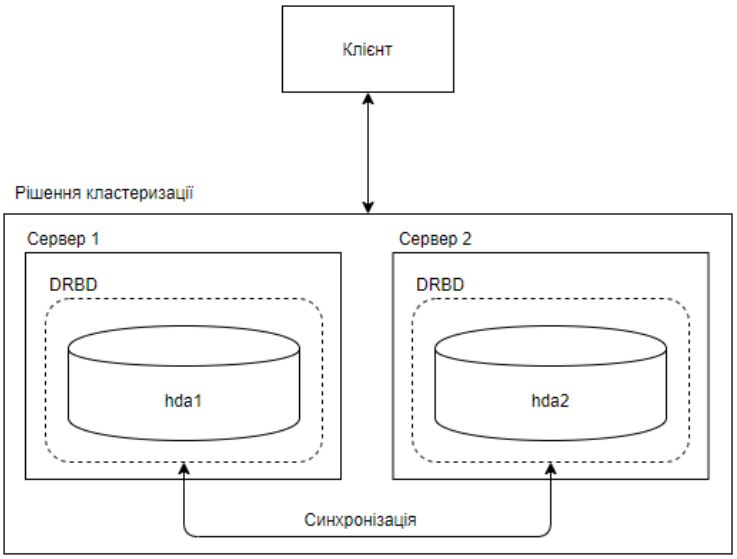


Рисунок 2.4 – Основна модель роботи DRBD

Основною моделлю роботи DRBD є активно/пасивна модель [16]. Основний вузол використовується для операцій читання і запису для всіх користувачів. Вторинний вузол стає первинним, якщо рішення кластеризації виявить, що первинний вузол відмовив. Операції запису відбуваються через первинний вузол і виконуються одночасно в локальному та вторинному сховищах (рисунок 2.5). DRBD підтримує два режими для операцій запису, які називаються синхронними та асинхронними.

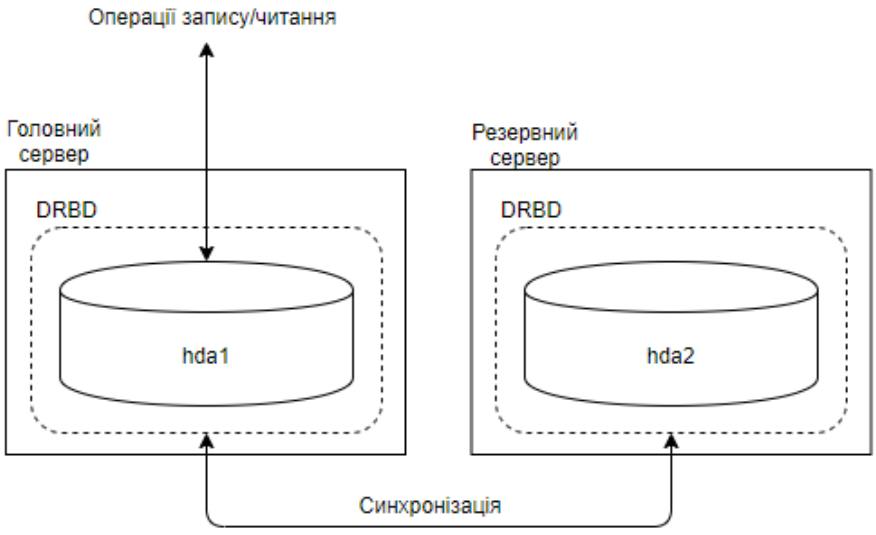


Рисунок 2.5 – Операції читання/запису з DRBD

У повністю синхронному режимі, операції запису повинні бути відіслані до обох вузлів, перш ніж транзакція буде підтверджена автором запису. В асинхронному режимі транзакція запису відбувається після збереження даних запису у сховищі локального вузла.

DRBD підтримує активну/активну модель, так, що операції читання і запису можуть відбуватися на обох серверах одночасно.

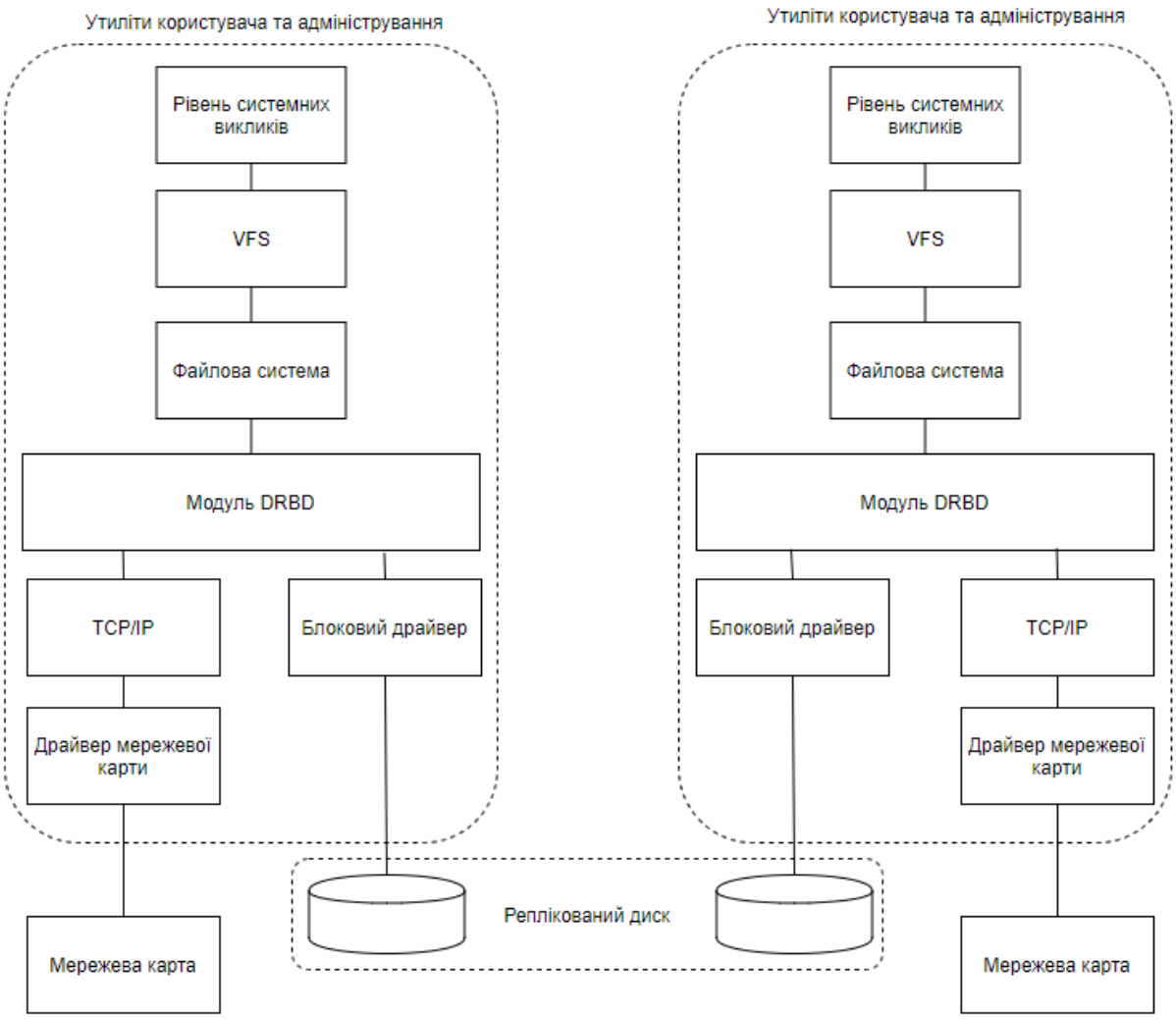


Рисунок 2.6 – DRBD в архітектурі Linux

DRBD можливо розділити на дві незалежні частини: модуль ядра, який реалізує поведінку DRBD, і набір програм адміністрування простору користувачів, що використовуються для управління дисками DRBD (рисунок 2.6).

Модуль ядра реалізує драйвер віртуального блокового пристрою. Модуль DRBD реалізує інтерфейс не тільки до основного драйвера блоку, але і мережевого стеку, кінцева точка, якого визначається IP-адресою та номером порту. DRBD має такий ряд переваг:

– має shared-secret автентифікацію;

– сумісний з LVM (Logical Volume Manager);

– існує підтримка інтеграції heartbeat/pacemaker;

– існує підтримка балансування для запитів на читання;

– автоматична синхронізація після відмови вузла;

– автоматичне управління пропускною здатністю;

– інтеграція з рішеннями для віртуалізації.

При використанні даної технології з великою кількістю вузлів, копія файлу буде зберігатися на кожному вузлі, при цьому розмір мережевого диску буде рівним найменшому диску вузла кластеру, що однозначно вплине на масштабованість та продуктивність файлової системи.

2.5.2 Підходи до забезпечення відмовостійкості бази даних

Обрана система проведення олімпіад – ejudge використовує базу даних MySQL для зберігання даних, саме тому для слід розглянути методи для забезпечення відмовостійкості бази даних.

2.5.2.1 Реплікація блоків даних

Кожен вузол має блок пристроїв, призначений для зберігання даних [17]. Один з них знаходиться в активному режимі і працює як сервер бази даних. Решта вузлів перебувають у режимі пасивного очікування – будь-які зміни, внесені на блоковому пристрої активного вузла, реплікуються на пасивні вузли.

Реплікація може бути синхронною, асинхронною або напівсинхронною [18]. У випадку відмови активного вузла пасивні вузли мають точну копію даних (рисунок 2.7). Спеціального налаштування вузлів не потребується бо усі екземпляри MySQL звертаються до спільного сховища.

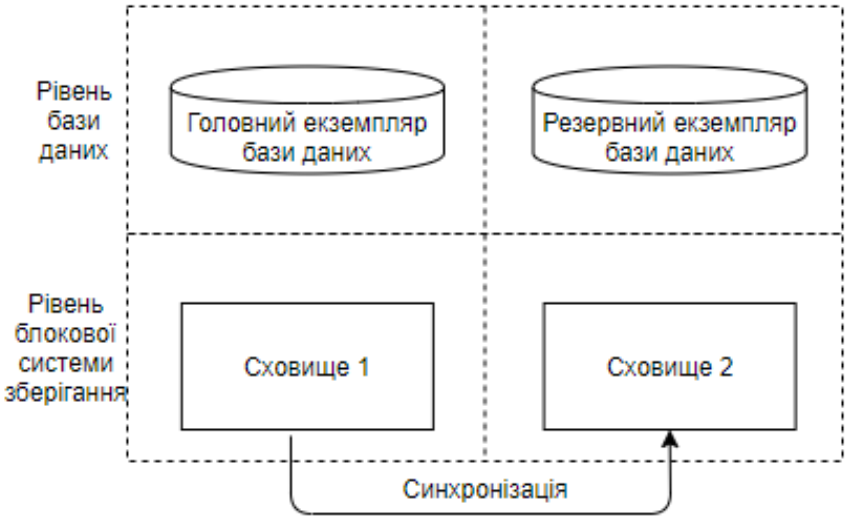


Рисунок 2.7 – Реплікація блоків даних

Даний підхід має певний ряд недоліків. Один з головних – активно-пасивний підхід. У кластері повинно бути мінімум два вузли, а використовувати можливо лише один. Відмова прирівнюється до запуску MySQL з пасивного стану, що може зайняти багато часу для відновлення системи.

2.5.2.2 Топологія Master with Slaves (Одинична реплікація)

Одинична реплікація - це найпростіша топологія реплікації MySQL (рисунок 2.8). Головний екземпляр отримує запит на запис та реплікує його за допомогою асинхронної або напівсинхронної реплікації, до допоміжних вузлів кластеру. У випадку відмови головного екземпляру, його роль перейде

до допоміжного. Решта допоміжних екземплярів поновлять свою роботу з новим головним вузлом.

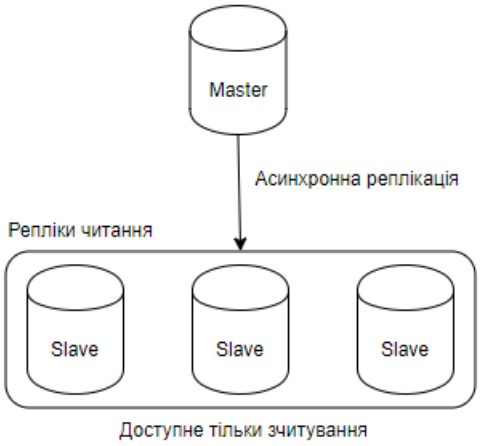


Рисунок 2.8 – Одинична реплікація

2.5.2.3 Топологія Master with Relay Slaves (Ланцюжкова реплікація)

У даній топології використовується проміжний вузол, щоб діяти як ретранслятор для допоміжних вузлів.

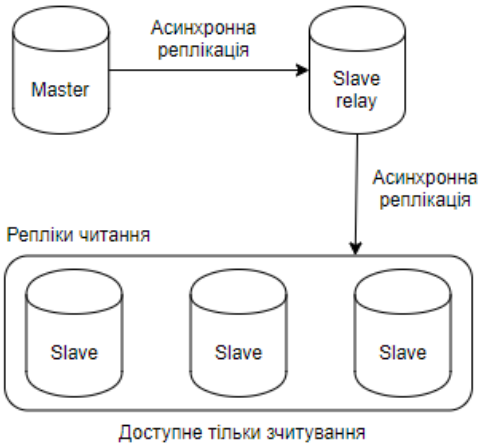


Рисунок 2.9 – Ланцюжкова реплікація

На рисунку 2.9 зображена топологія ланцюжкової реплікації. Якщо до ведучого підключено багато допоміжних вузлів, мережевий інтерфейс ведучого може перевантажуватися. Проміжний вузол використовується для

зменшення навантаження для операцій з читання. Використання проміжного вузла має певні недоліки:

– затримка реплікації на підлеглому сервері призведе до затримки реплікації на підлеглих вузлах;

– несанкціоновані транзакції на сервері проміжного вузла будуть виконуватися на допоміжних;

– якщо проміжного сервер виходить з ладу, і не використовується GTID, всі підлеглі перестають виконувати реплікацію і їх потрібно повторно ініціалізувати.

2.5.2.4 Топологія Master with Backup Master (Множинна реплікація)

Головний вузол висуває зміни до резервного майстра та до одного або декількох допоміжних. Напівсинхронна реплікація використовується між головним і резервним вузлом та впливає на продуктивність кластеру, але ризик втрати даних зводиться до мінімуму.

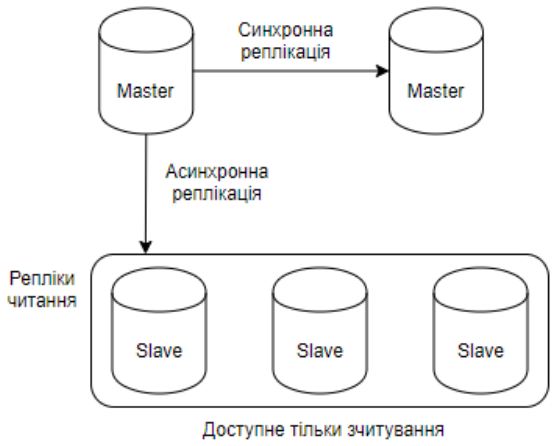


Рисунок 2.10 – Множинна реплікація

Дана топологія (рисунок 2.10) може використовуватися для забезпечення відмовостійкості головного вузлу. Майстер резервного копіювання діє як сервер гарячої готовності, оскільки він має найбільшу

ймовірність наявності актуальних даних у порівнянні з допоміжними екземплярами.

2.5.2.5 Топологія Multiple Masters to Single Slave (Реплікація з кількома джерелами)

Реплікація з кількома джерелами (рисунок 2.11) дозволяє головному вузлу реплікації одночасно приймати транзакції з декількох джерел. Реплікація з кількома джерелами може використовуватися для об'єднання фрагментів таблиці та консолідації даних з декількох серверів на одному сервері.

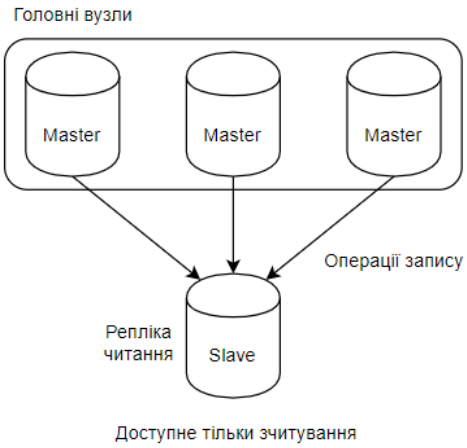


Рисунок 2.11 – Реплікація з кількома джерелами

MySQL та MariaDB мають різні реалізації реплікації з кількома джерелами, де MariaDB повинен мати GTID, в той час як MySQL використовує окремий канал реплікації для кожного майстра, який веде реплікації підлеглого.

2.5.2.6 Топологія гібридної реплікації

Гібридна реплікація (рисунок 2.12) – це поєднання асинхронної реплікації MySQL і синхронної реплікації, що надається Galera. Дана

топологія принципово відрізняється від стандартної реплікації MySQL і вирішує ряд проблем, включаючи конфлікти під час запису на кількох вузлах Master, проблему з відставанням реплікації, яка веде до десинхронізації ведених вузлів з майстром.

У кластері Galera додаток може виконувати запис на будь-який вузол, а фіксації транзакцій, або подій реплікації на основі рядків (RBR), застосовуються до всіх серверів за допомогою реплікації на основі сертифікації. Реплікація на основі сертифікації – це альтернативний підхід до синхронної реплікації бази даних з використанням методів групової взаємодії та впорядкування транзакцій.

Мінімальний кластер Galera складається з 3 вузлів. Рекомендується використання кластеру з непарною кількістю вузлів, щоб при виникненні проблем із застосуванням транзакції на одному вузлі два інших вузла могли створити кворум і продовжити виконання транзакції. Дана схема ілюструє відмінності між реплікацією MySQL та кластером Galera.

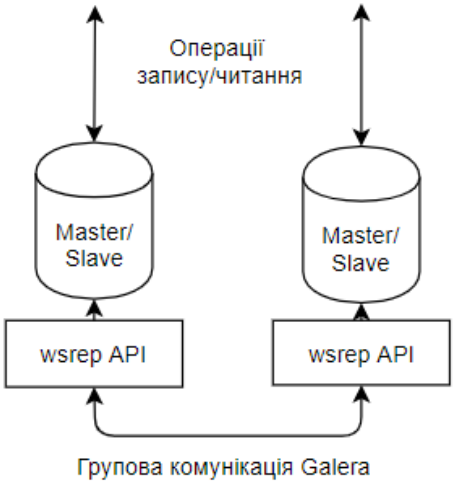


Рисунок 2.12 – Реплікація galera

Galera Cluster реалізує реплікацію, використовуючи такі компоненти:

– система управління базами даних – сервер бази даних, який працює на окремому вузлі. Підтримувані СУБД є MySQL Server, Percona Server для MySQL та MariaDB Server;

– wsrep API – інтерфейс для серверу бази даних та постачальник реплікацій. Інтерфейс забезпечує інтеграцію з сервером бази даних для реплікації набору записів;

– плагін Galera – плагін, який дозволяє встановити функцію обслуговування реплікації;

– плагіни групової комунікації – системи групового зв’язку, доступні кластеру Galera.

Постачальник, який вбудовує технологію Galera Cluster в СУБД, повинен включити в кодову базу сервера СУБД патч WriteSet Replication (WSRep) [19]. Це дозволить розширенню Galera, що працює в якості провайдера WSRep, взаємодіяти і реплікувати транзакції за допомогою протоколу групового зв'язку, що забезпечує роботу синхронної мульти-майстер реплікації для InnoDB. В результаті транзакції виконуються синхронно на всіх вузлах.

Якщо один з вузлів вийшов з ладу, інші вузли будуть продовжувати працювати і відстежувати зміни даних. Коли вузол відновлюється, перед поверненням в кластер, він автоматично синхронізується з іншими вузлами за допомогою передачі знімка стану (SST) або інкрементної передачі стану (IST) в залежності від останнього відомого стану. При збої будь-якого з вузлів дані не втрачаються.

Завдяки гібридній реплікації забезпечується відмовостійкість усіх вузлів кластеру бази даних.

2.6 Вимоги до підсистеми енергоживлення

Надійність підсистеми живлення є важливою складовою відмовостійкості кластеру. Вимогою для побудови відмовостійкої системи є наявність резервної лінії електроживлення, що подається за окремим кабелем [20].

При використанні такої схеми (рисунок 2.13) існує необхідна надмірність, що дозволяє запобігти простою при одноразовому збою в системі розподілу живлення. Така надмірність також полегшує обслуговування системи електроживлення.

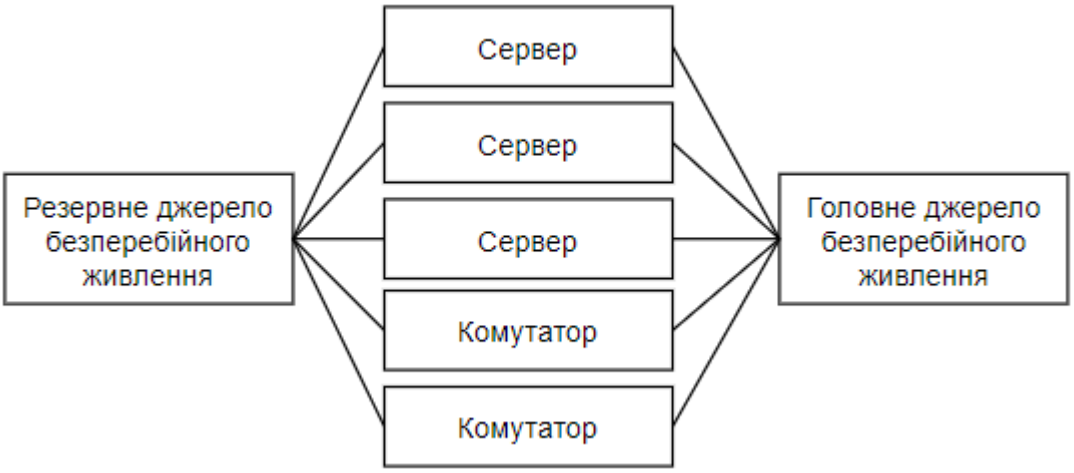


Рисунок 2.13 – Надмірність системи живлення

Відмовостійкість мережевої підсистеми повинна забезпечуватися комбінацією рішень наступних завдань:

– відмовостійкість з'єднань;

– відмовостійкість мережевих пристроїв.

Для здійснення агрегування в мережах Ethernet розроблений протокол LACP (Link Aggregation Control Protocol). Агреговані канали LACP використовуються як для підвищення пропускної здатності, так і для забезпечення відмовостійкості. Описується стандартом IEEE 802.3ad

Відмовостійкість комутаторів забезпечується паралельною роботою обчислювальних вузлів з двома комутаторами (рисунок 2.14). На малюнку зображена конфігурація сервер-комутатор, яка забезпечує відмовостійкість у тому випадку, якщо збій відбувається на комутаторі, однак при цьому не забезпечується агрегування з'єднання і балансування навантаження.

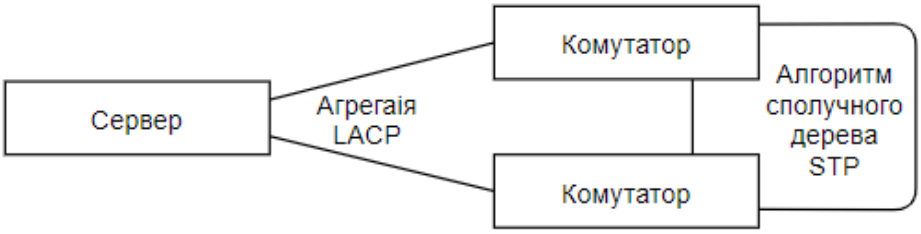


Рисунок 2.14 – Надмірність з’єднань

Для такої конфігурації потрібно задіяти алгоритм сполучного дерева (STP) на обох комутаторах, що дозволить гарантувати, що тільки одне з'єднання буде активним.

Для створення мінімальної відмовостійкої конфігурації необхідна одна мережа, що включає два комутатора. Однак така конфігурація має ряд серйозних недоліків:

– мережевий обмін прикладних задач може негативно позначатися на швидкості роботи поділюваних сховищ даних;

– мережеві збої прикладних програм можуть призвести до порушень функціонування програмного забезпечення управління кластером;

– неможливість використання IP-адрес в якості керованого ресурсу кластера.

Подолання цих проблем можливе шляхом сегментації та ізоляції трьох функціональних підмереж: прикладної, управління, сховища даних. Сегменти мережі потрібно ізолювати фізично – використовувати незалежні мережі з виділеним мережевим обладнанням. У разі неможливості фізичного ізолювання необхідно використовувати логічний поділ на основі VLAN, з налаштуванням QoS для сегментів.

2.7 Розробка схеми резервування фізичного рівня кластеру

Виходячи з вимог до підсистеми енергоживлення та резервування каналів зв’язку створена схема, яка зображена на кресленику

ІА62.010БАК.005 Д1. На даній схемі зображене забезпечення відмовостійкості підсистеми живлення та мережевої підсистеми.

Забезпечення відмовостійкості підсистеми живлення забезпечується використанням основного джерела безперперебійного живлення та резервуванням джерела живлення. На схемі зображені вузли кластеру та мережеве обладнання, які мають два блоки живлення та під'єднані до різних джерел живлення. Мережеве та серверне обладнання зазвичай використовують лише головний блок живлення, допоміжний починає використовуватися при відмові головного. Резервування живлення обладнання забезпечує високу відмовостійкість апаратної частини кластеру. При виході з ладу блоку живлення, або джерела безперебійного живлення обладнання одразу почне використовувати резервне джерело живлення, при цьому продовжить працювати за призначенням без переривань.

Забезпечення відмовостійкості мережевої підсистеми забезпечується резервуванням мережевих з'єднань. Кожен вузол має два мережевих інтерфейси, які під'єднані до двох різних комутаторів та об'єднані за допомогою протоколів LACP та STP. При виході з ладу комутатора, порта комутатора, фізичного інтерфейсу, кабелю, вузол кластеру буде використовувати резервний мережевий інтерфейс, що забезпечить безперебійне логічне з'єднання вузла до мережі кластеру.

На схемі також зображена підсистема контролю вузлом, яка не потребує резервації, бо вона використовується тільки при встановленні операційної системи, оновленні системи, контролю енергоживлення. Вихід з ладу підсистеми контролю не вплине на роботу вузла, тому не має сенсу забезпечувати її відмовостійкість та використовувати резервування її інтерфейсів.

Дана схема використовується для забезпечення відмовостійкості апаратної частини кластеру розроблюваної системи. Резервування забезпечує роботу обладнання у разі відмови одного з компонентів розглянутих підсистем.

2.8 Розробка структурної сервісної архітектури тестуючої системи

Тестуюча система ejudge складається з таких основних модулів:

– керуючі компоненти;

– CGI-програми;

– допоміжні програми.

Керуючі компоненти – відповідають за доступ до основної бази даних користувачів, моніторинг активних турнірів, виконання фонових допоміжних завдань і компіляцію рішень учасників [21]. Керуючі компоненти системи працюють весь час роботи системи, починаючи від створення турніру, реєстрації користувачів, проведення турніру Керуючі компоненти працюють в одному екземплярі для всіх існуючих в системі турнірах. Запуск системи ejudge полягає в запуску керуючих компонент. До цієї групи належать такі програми:

CGI-програми відповідають за безпосередню обробку запитів користувачів і адміністраторів системи та надають інтерфейс до програм першої і другої груп. CGI-програми запускаються веб-сервером під час запиту від клієнта. Вони виконують попередній аналіз запиту і передають запит на генерацію HTML-сторінок відповідними програмами серверної частини: ej-users, ej-super-server, ej-contests.

Допоміжні програми, які надають доступ до функцій системи з командного рядка і виконують інші функції. Вони запускаються з командного рядка і призначені для використання адміністратором системи ejudge.

Архітектура тестуючої системи ejudge представлена у вигляді схемі ІА62.010БАК.005 Д2. Тестуюча система має 4 головних рівня:

– відображення (web інтерфейс);

– контролю (екземпляр ejudge);

– зберігання (база даних, файлова система);

– тестування (виконання та перевірка надісланих програм) .

Рівень відображення відповідає за відображення сторінок кінцевим користувачам системи. Головна функція рівня відображення – вибірка даних з рівня контролю, перетворення їх до вигляду сторінок та передача кінцевим користувачам системи.

Рівень контролю відповідає за:

– безпосередній контроль над тестуючою системою;

– формування запитів для виконання компіляції та тестування на рівні тестування;

– відправка до рівню відображення набору даних щодо статусу тестування та компіляції програм;

– вибірка даних з рівня даних для формування набору даних про статус тестування та компіляції.

Рівень тестування відповідає за компіляцію та тестування отриманого коду з рівня контролю та подальше занесення даних до бази даних та зберігання вхідних та вихідних файлів у файловій системі. Рівень складається з тестуючого та компілюючого модулю.

Рівень даних відповідає збереження даних для рівня контролю та рівня тестування. Рівень складається з бази даних та файлової системи.

Дана архітектура надає системі ознаку модульності, що відповідає вимогам до проектування додатків за дозволяє використання системи для створення відмовостійкого кластеру.

Для забезпечення високої надійності тестуючої системи, необхідно аби дані рівні були стійкі до відмов. Для забезпечення стійкості бази даних слід використовувати гібридну реплікацію, яка гарантує відмовостійкість ведучих та допоміжних вузлів, та гарантує коректну роботу та відновлення бази даних при відмові більшості вузлів кластеру. Для забезпечення безперебійної роботи файлової системи найкращим інструментів є Ceph. В порівнянні з

HDFS та DRBD Ceph не має єдиної точки відмови та не має обмежень в розмірі файлів.

Для забезпечення стійкості до відмов рівня відображення можна доповнити систему за допомогою відмовостійкої системи балансування трафіку користувачів між доступними серверами за допомогою CDN. Вузли, які вийшли з ладу автоматично будуть усуватися з балансування. Виконання цього кроку не тільки забезпечує автоматичне відключення неактивних серверів, але і дозволяє забезпечити захист кластера від DDoS.

Після побудови відмовостійкого кластеру маємо підсумкову схему сервісів ІА62.010БАК.005 Д3 для тестуючої системи ejudge. На даній схемі зображена взаємодія сервісів у створеному кластері. Екземпляри бази даних утворюють між собою кластер бази даних, що забезпечує відмовостійкість бази даних. Екземпляри розподіленої файлової системи утворюють між собою розподілену файлову систему, що забезпечує відмовостійкість файлової системи. Кластер повинен мати декілька мереж, для того, щоб розділяти мережі даних клієнтів та трафіку системи даних кластеру. Більшість часу мережевий обмін в середовищі кластеру перевищує за обсягом обмін даними клієнтів. З цього випливає доцільність створення окремих мереж:

– мережа бази даних;

– мережа розподіленого файлового сховища;

– мережа WAN;

– локальна мережа кластеру.

Наступний розділ присвячений вирішенню даних питань при безпосередньому розгортанні системи.

ВИСНОВОК ДО РОЗДІЛУ 2

В якості основи для розроблюваної обрана тестуюча система ejudge. Тестуюча система використовує базу даних MySQL, для забезпечення відмовостійкості якої, використана топологія гібридної реплікації завдяки таким перевагам:

– фіксація транзакції та реплікація на основі сертифікації – ці властивості дозволяють забезпечити цілісність даних, при втраті вузла, запит автоматично надсилається до іншого або клієнту повертається відповідь з відмовою;

– відсутність спеціалізації вузлів – на відміну від інших підходів Galera не потребує конфігурування окремих типів вузлів, вибір та розподілення операцій контролюється за допомогою екземпляру Galera на кожному вузлі кластеру.

Відповідно до вимог надійності вузла система повинна мати загальну файлову систему. На основі проаналізованих розподілених файлових систем обрана система Ceph завдяки таким перевагам:

– розділення файлу на блоки – це дозволяє розподілити копії блоків файлу на вузлах кластеру, що забезпечує високу відмовостійкість при відмові одного з вузлів з блоком. При збої вузла виконається реплікація блоку до іншого вузла;

– автоматичне відновлення – кластер постійно контролюється, якщо одна із триразових копій відсутня, копія створюється автоматично, щоб гарантувати наявність трьох примірників;

Розроблені графічні матеріали за допомогою яких можливе створення кластеру. Безпосереднє створення кластеру та розгортання системи розглянуто у розділі 3.

3 РОЗГОРТАННЯ ТЕСТУЮЧОЇ СИСТЕМИ

3.1 Встановлення операційної системи

Для запуску образу установки необхідно попередньо записати образ системи Ubuntu на флеш накопичувач або DVD диск, налаштувати BIOS вибравши пристрій у меню «BOOT» та перезавантаживши сервер. При запуску серверу почнеться ініціалізація системи, в меню вибору необхідно вибрати «Встановлення». Потрібно вибрати мову системи, розкладку клавіатури, регіон. Інсталятором буде рекомендовано налаштувати мережу (рисунок 3.1)

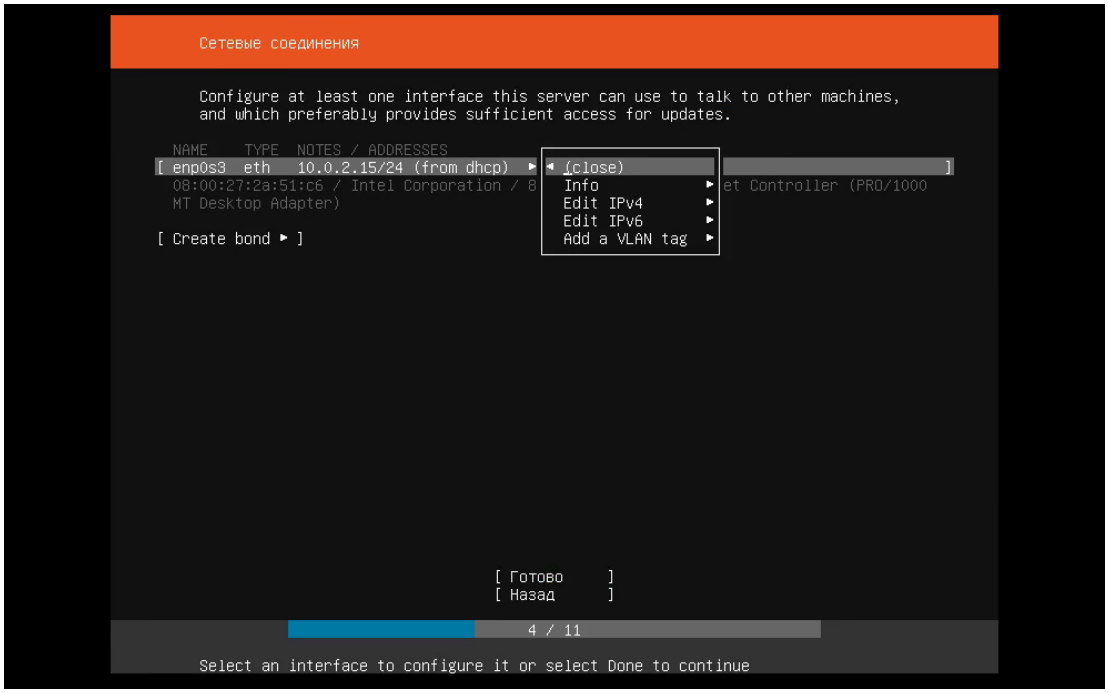


Рисунок 3.1 – Налаштування мережі

Рекомендується виконати налаштування підмережі з доступом до мережі інтернет, подальше налаштування мережі описано у розділі «Налаштування мережі кластеру». Одним із найважливіших налаштувань є вибір файлової системи. Необхідно вибрати параметр «Use an entire disk» та обрати один із дисків для встановлення операційної системи (рисунок 3.2).

Інший диск буде налаштовуватися та використовуватися файловою підсистемою Ceph.

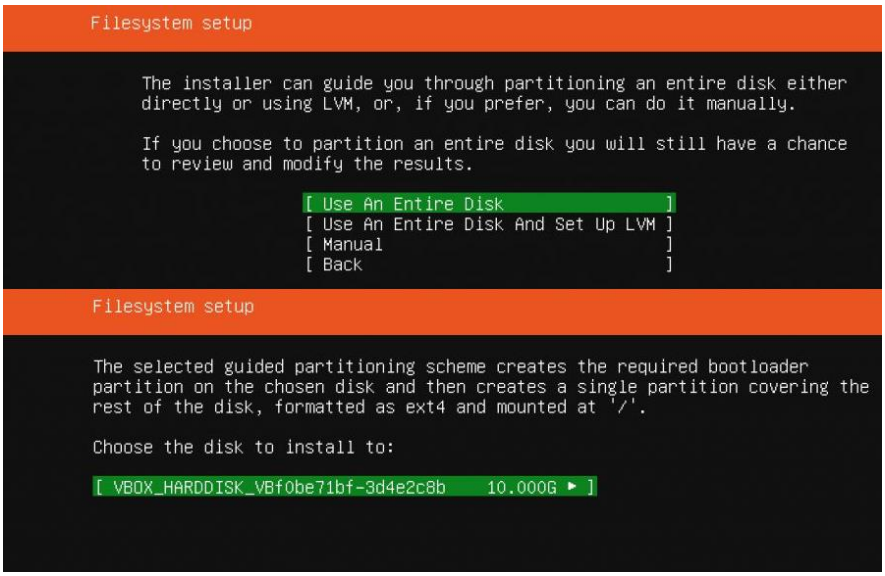


Рисунок 3.2 – Налаштування дисків

Інсталятор запропонує створити користувача серверу після збереження налаштувань дисків (рисунок 3.3). Рекомендується називати вузли однаковим іменем: node1, node2.

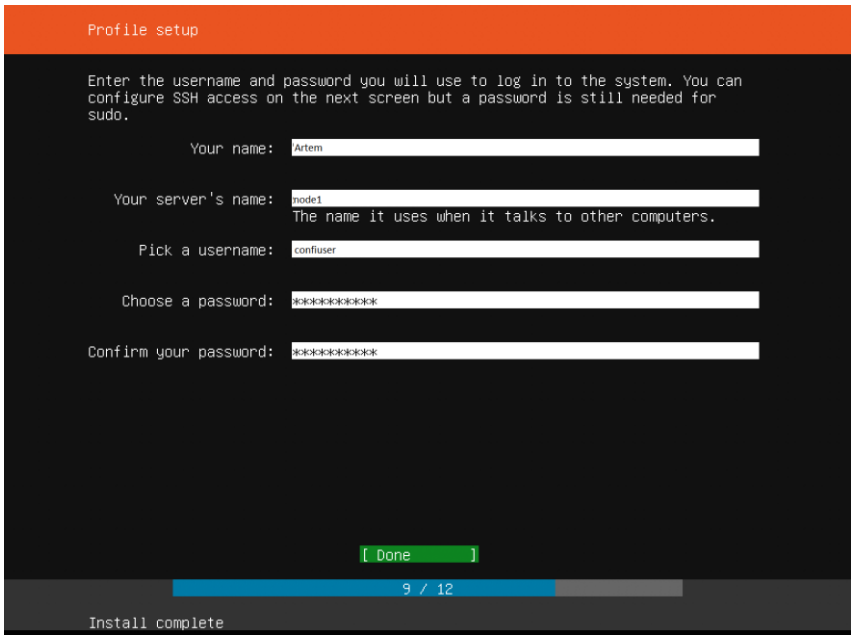


Рисунок 3.3 – Налаштування користувачів системи

Виконавши встановлення системи сервер перезавантажиться. Після перезавантаження необхідно ввести дані користувача та оновити пакети системи до останньої версії – це можливо зробити за допомогою команди «sudo apt update && sudo apt full-upgrade». Виконання даних команд підготовить сервер до подальшого налаштування.

3.2 Налаштування мережі кластеру

Кожен вузол кластеру повинен мати окрему підмережу для бази даних, мережевого сховища, зовнішню з виходом в Інтернет. Для забезпечення даної конфігурації слід використовувати VLAN , який дозволяє створити декілька локальних мереж на одному фізичному інтерфейсі. Дану технологію використовують для створення логічної топології мережі. Для кластеру необхідно створити 4 локальні мережі та присвоїти їм ідентифікатор VLAN. Виходячи з мінімальних характеристик вузол повинен мати 2 фізичних мережевих інтерфейси, які повинні бути сконфігуровані як один логічний інтерфейс за допомогою протокол LACP.

Для об’єднання фізичних інтерфейсів та створення VLAN необхідно відредагувати файл /etc/network/interfaces (рисунок 3.4).

Для іншиих вузлів необхідно виконати попередні кроки з налаштування мережі. На комутаторах необхідно налаштувати VLAN та протокол зв`язуючого дерева для забезпечення резервування фізичних з’єднаннь. Для кластеру обрані комутатори компанії Cisco – SG350X. Для їх налаштування необхідно авторизуватися до комутатору, зайти в привілейований режим та виконати команду «spanning-tree». Наступним кроком необхідно сконфігурувати STP протокол та вибрати необхідні параметри, а саме:

а) stp – Класичний STP забезпечує єдиний шлях між будь-якими двома кінцевими точками, виключаючи та запобігаючи петлям мереж;

б) rstp – RSTP виявляє мережеві топології для забезпечення швидшого зближення дерева, що охоплює. Ця опція включена за замовчуванням;

в) mst – MSTP заснований на RSTP. Параметр виявляє петлі другого рівня та намагається пом'якшити їх, не даючи причетному порту передавати трафік.

```
auto eth0
iface eth0 inet manual
bond-master bond0

auto eth1
iface eth1 inet manual
bond-master bond0


auto bond0
iface bond0 inet static

        address 192.168.1.1
        netmask 255.255.255.0
        bond-mode 802.3ad
        bond-miimon 100
        bond-slaves eth0 eth1

auto vlan10
iface vlan10 inet static
        address 192.168.10.1
        netmask 255.255.255.0
        broadcast 192.168.10.255
        vlan-raw-device bond0

auto vlan11
iface vlan11 inet static
        address 192.168.11.1
        netmask 255.255.255.0
        broadcast 192.168.11.255
        vlan-raw-device bond0

auto vlan12
iface vlan12 inet static
        address 192.168.12.1
        netmask 255.255.255.0
        broadcast 192.168.12.255
        vlan-raw-device bond0
```

Рисунок 3.4 – Мережева конфігурація вузла

Для даної архітектури обраний параметр a– який дозволить зупинити мережевий флуд, при випадковому петлюванні в мережі. Після налаштування STP необхідно налаштувати VLAN для кожного порту, до якого підключений вузол кластеру (рисунок 3.5).

```
!vlan 10 is my storage VLAN
interface Port-channel1
 switchport trunk allowed vlan 10
 switchport mode trunk
end
!vlan 11 is my data VLAN
interface Port-channel1
 switchport trunk allowed vlan 12
 switchport mode trunk
end

!vlan 12 is my inet VLAN
interface Port-channel1
 switchport trunk allowed vlan 13
 switchport mode trunk
end

interface GigabitEthernet0/43
 switchport trunk allowed vlan 10
 switchport trunk allowed vlan 11
 switchport trunk allowed vlan 12
 switchport mode trunk
 channel-group 1 mode active
end
```

```
interface GigabitEthernet0/44
  switchport trunk allowed vlan 10
  switchport trunk allowed vlan 11
  switchport trunk allowed vlan 12
  switchport mode trunk
 channel-group 1 mode active
end


interface GigabitEthernet0/45
  switchport trunk allowed vlan 10
  switchport trunk allowed vlan 11
  switchport trunk allowed vlan 12
  switchport mode trunk
 channel-group 1 mode active
end

interface GigabitEthernet0/46
  switchport trunk allowed vlan 10
  switchport trunk allowed vlan 11
  switchport trunk allowed vlan 12
  switchport mode trunk
 channel-group 1 mode active
end
```

Рисунок 3.5 – Приклад налаштування комутатора cisco

3.3 Конфігурація кластеру бази даних

Кожен вузол повинен мати назви вузлів та їх відповідність IP адресу. Для вузла треба змінити файл /etc/hosts (рисунок 3.6). – який відповідає за початкове визначення доменних імен.

```
192.168.10.1    galera-db-01
192.168.10.2    galera-db-02
192.168.10.3    galera-db-03
```

Рисунок 3.6 – Приклад налаштування файлу /etc/hosts

Наступним кроком необхідно перевірити доступність вузлів за допомогою команди «ping -c2 galera-db-01». Для налаштування galera необхідно встановити такі компоненти: mariadb-server та rsync [22]. На першому вузлі необхідно виконати таку конфігурацію у файлі /etc/mysql/mariadb.conf.d/galera.cnf (додаток А.1). Для інших вузлів файл конфігурації наведений у додатку А.2 та додатку А.3.

Іншим кроком слід зупинити сервіс mariadb командою «systemctl stop mariadb» на усіх вузлах кластеру. На першому вузлі слід запустити кластер Galera командою «galera_new_cluster». Після старту Galera необхідно перевірити стан, виконавши команду «mysql -u root -p -e "show status like 'wsrep_cluster_size'"», яка повинна повернути параметр wsrep_cluster_size рівним одиниці. Наступним кроком необхідно запустити MySQL на інших вузлах кластеру командою «systemctl start mariadb». Для остаточної перевірки необхідно створити тестову базу даних на будь-якому вузлі кластеру. Після створення бази даних необхідно виконати перевірку на наявність на іншому вузлі кластеру за допомогою команди «show databases;» (рисунок 3.7).

```
root@galera-node-01:~# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 50
Server version: 10.3.7-MariaDB-1:10.3.7+maria~bionic-log mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database galera_test;
Query OK, 1 row affected (0.004 sec)
```

Рисунок 3.7 – Приклад створення бази даних

3.4 Конфігурація розподіленої файлової системи

Кожен вузол повинен мати назви вузлів та їх відповідність IP адресу. Для вузла треба змінити файл /etc/hosts (рисунок 3.8) – який відповідає за початкове визначення доменних імен.

```
192.168.11.1 ceph-node-00
192.168.11.2 ceph-node-01
192.168.11.3 ceph-node-03
```

Рисунок 3.8 – Приклад налаштування файлу /etc/hosts

Після зміни файлу конфігурації необхідно перевірити доступність вузлів за допомогою команди «ping -c2 galera-db-01».

Для використання Ceph на вузлі кластеру необхідно створити користувача та надати йому права доступу «sudo», додавши запис до файлу /etc/sudoers на кожному сервері [23]. Після створення користувача «ceph» необхідно створити та розповсюдити до інших вузлів rsa ключ для безпарольної взаємодії між вузлами кластеру. Це можливо зробити за допомогою команди «ssh-copy» (рисунок 3.9). Дана команда скопіює приватний та публічний ключ до іншого вузла кластеру та наддасть безпарольний доступ.

```
ghhf@ceph-node-01:~/$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ghhf/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ghhf/.ssh/id_rsa.
Your public key has been saved in /home/ghhf/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:3NjXSK9e+xUSI4LoE059Xed7nEU3lMy0mhFfzsTjPFU ghhf@KV-LT-9280Y2J
The key's randomart image is:
+---[RSA 2048]----+
|             o=*E|
|      o . . . =XB|
|     + o o o.+o+*|
|    + ...+...+*=+|
|     +  S o o+o++|
|      .    . .. o|
|            . . .|
|           . . ..|
|            . ...|
+----[SHA256]-----+
ghhf@ceph-node-01:~/$ ssh-copy-id ceph@ceph-node-00
ghhf@ceph-node-01:~/$ ssh-copy-id ceph@ceph-node-01
ghhf@ceph-node-01:~/$ ssh-copy-id ceph@ceph-node-02
```

Рисунок 3.9 – Створення та копіювання ssh ключу

Для коректної роботи файлової системи необхідно встановити наступні програми та компоненти: ceph-deploy, ntp, lvm. Для початку встановлення

Ceph необхідно створити файлову директорію «ceph-deploy» використовуючи створеного користувача «ceph». Дана директорія призначена для зберігання тимчасових файлів, які будуть створені під час встановлення. Для створення кластеру необхідно виконати команду «ceph-deploy» (рисунок 3.10).

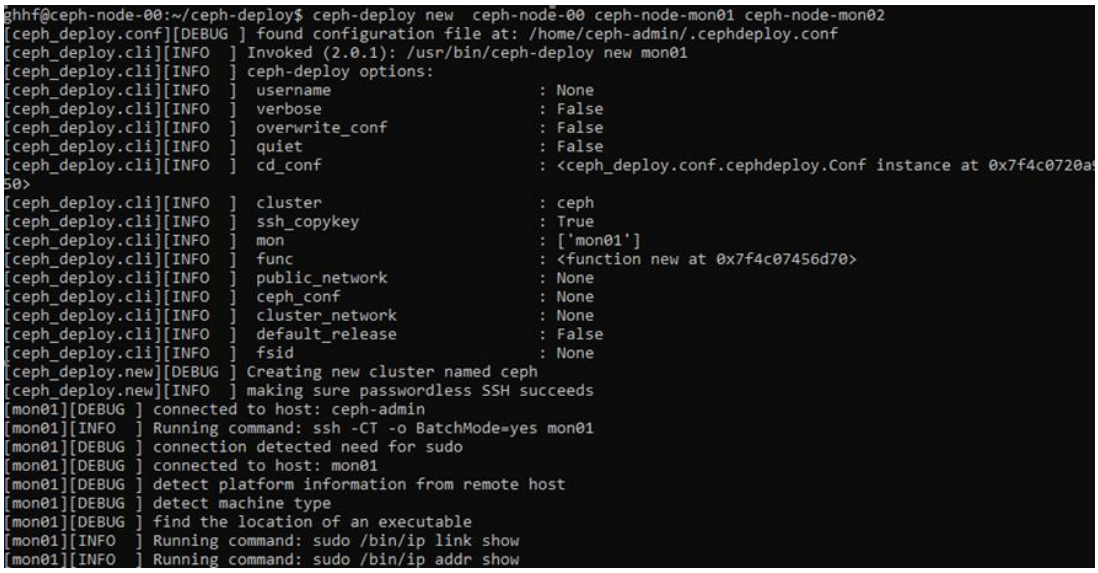


Рисунок 3.10 – Створення кластеру ceph

Наступним кроком слід змінити кількість реплік файлів. За замовчуванням значення «osd_pool_default_min_size» рівне 3, для збільшення загального файлового сховища слід зменшити значення до 2 у створеному файлі ceph.conf. Після перевірки та модифікації файлу потрібно встановити необхідні компоненти та оновити конфігурації на вузлах кластеру за допомогою команди «ceph-deploy install».

Для забезпечення високої доступності необхідно запустити Ceph кластер як мінімум з трьома MON. Ceph використовує алгоритм, який вимагає консенсусу серед MON в кворумі. Для запуску Ceph MON слід виконати дану команду «ceph-deploy mon create-initial» на усіх вузлах, після виконання, на кластері будуть сконфігуровані додатки MON кластеру.

При встановленні операційної системи задіяний лише один фізичний диск (/dev/sda), на даний момент конфігурація та розмітка файлової системи

диску повинна бути відсутньою. Необхідно виконати початкову підготовку диску – розмітка файлової системи, конфігурація необхідних компонентів. Для цього слід виконати команду «ceph-deploy osd prepare ceph-node-00:sdb ceph-node-01:sdb ceph-node-02:sdb». Після підготовки дисків треба активувати OSD при цьому вказавши необхідні розділи дисків. Зробити це можна за допомогою команди «ceph-deploy osd activate ceph-node-00:sdb1:/dev/sdb2 ceph-node-01:sdb1:/dev/sdb2 ceph-node-02:sdb1:/dev/sdb2».

Для перевірки успішності інсталяції потрібно виконати команду «ceph status» (рисунок 3.11).

```
    cluster fb82947c-7b0d-4b7d-8bd9-c0386f48c152
     health HEALTH_WARN
            64 pgs stuck inactive
            64 pgs stuck unclean
            too few PGs per OSD (16 < min 30)
     monmap e1: 1 mons at {ceph-node-mon=192.168.2.68:6789/0}
            election epoch 1, quorum 0 ceph-node-mon
     osdmap e13: 4 osds: 4 up, 4 in
            flags sortbitwise
      pgmap v18: 64 pgs, 1 pools, 0 bytes data, 0 objects
            130 MB used, 61265 MB / 61395 MB avail
                  64 creating
```

Рисунок 3.11 – Перевірка коректності інсталяції кластеру ceph

Для подальшого використання файлової системи слід створити точку монтування у системі та виконати монтування файлової системи (рисунок 3.12).

```
mkdir -p /etc/ceph
echo "AQBdX0NXobyACBAArLIkcDRrAHYEDc3izToMnA==" > /etc/ceph/cephfskey
sudo mkdir /mnt/cephfs
mount -t ceph 192.168.11.1:/ /mnt/cephfs/ -o name=cephfs,secretfile=/etc/ceph/cephfskey
```

Рисунок 3.12 – Монтування файлової системи до теки операційної системи

3.5 Встановлення тестуючої системи

Тестуюча система ejudge використовує LAMP стек для своєї роботи. Даний стек складається з таких компонентів:

– Linux – операційна система;

– Apache – веб сервер;

– MySQL – база даних ;

– PHP – мова програмування, за допомогою якої створюються веб-ресурси.

У розділі 3.1 та 3.3 розглянуто налаштування операційної системи та бази даних, відповідно. Для повноцінної роботи веб-додатку з використанням стеку LAMP необхідно встановити веб сервер, який буде відповідати за обробку запитів користувача. Для цього слід встановити пакет серверу Apache за допомогою команди «sudo apt install apache2».

Система використовує php в якості мови бекенду додатку. Для коректної роботи тестуючої системи на сервер необхідно інсталювати наступні пакети:

– libapache2-mod-php7.0;

– php7.0;

– php7.0-common;

– php7.0-curl;

– php7.0-dev;

– php7.0-gd;

– php-pear;

– php-imagick;

– php7.0-mcrypt;

– php7.0-mysql;

– php7.0-ps;

– php7.0-xsl.

Після інсталяції веб серверу необхідно відредагувати файл конфігурації веб серверу (додаток А.4) та (додаток А.5). Для забезпечення безпечної передачі даних між сервером та клієнтом слід забезпечити шифрування

запитів та відповідей від серверу. Для цього необхідно використовувати протокол HTTPS, який є безпечною модифікацією протоколу HTTP.

Протокол HTTPS особливо важливий у незахищених мережах (таких як публічні Wi-Fi точки доступу), оскільки будь-хто в локальних мережах може аналізувати трафік та перехоплювати чи змінювати інформацію, не захищену HTTPS [24]. Це означає, що гіпотетичний зловмисник може потенційно красти приватні дані користувача, отримувати доступ до облікового запису, вставляти шкідливий програмний код чи посилання на програмне забезпечення у сторінки, що надсилаються користувачеві у відповідь на його запити.

Одним із сертифікованих центрів видачі сертифікатів є центр Let`s encrypt. Даний центр має власне програмне забезпечення, яке дозволяє отримати приватний та публічний ключ, підписаний та сертифікований до використання у глобальній мережі. Для отримання сертифікату кожен вузол повинен мати зовнішню адресу та відповідний DNS запис. Підпис сертифікатів потребує встановлення наступних компонентів до вузла: certbot python-certbot-apache. Після встановлення необхідно виконати команду «sudo certbot --apache» (рисунок 3.13).

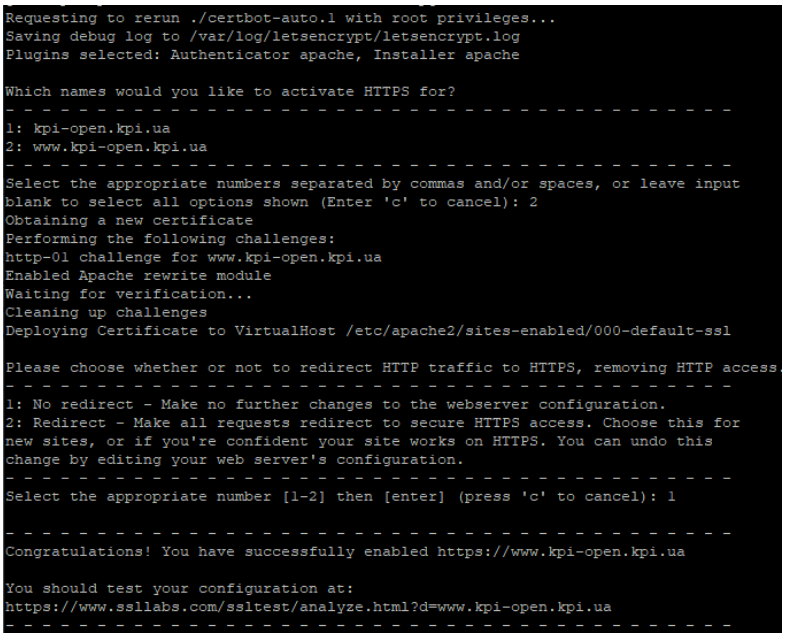


Рисунок 3.13 – Отримання ssl сертифікату

Сертифікат буде автоматично встановленний та доданий до конфігурацій. Тестуюча система ejudge потребує встановлення даних компонентів та компіляторів системи Linux:

Основні компоненти, які необхідні для запуску системи – sendmail, ncurses-base, libncurses-dev, libncursesw5, libncursesw5-dev, expat libexpat1, libexpat1-dev, zlib1g-dev, libelf-dev, awk, gawk, gettext, fpc, mc, libcurl4-openssl-dev, libzip-dev, uuid-dev, bison, flex [21].

Компілятори – необхідні для безпосередньої компіляції надісланого коду учасниками олімпіади:

– g++ – компілятор для коду, написаному на мові C++;

– openjdk-7-jdk – компілятор для коду, написаному на мові JAVA;

– mono-devel mono-runtime mono-vbnc – компілятор та компоненти, необхідні для перевірки програм, написаних на мові C# та VisualBasic;

– perl – компілятор для коду, написаному на мові Perl;

– python, python3 – компілятор для коду, написаному на мові Python;

– ruby – компілятор для коду, написаному на мові Ruby;

– gccgo – компілятор для коду, написаному на мові Go;

– gcc – компілятор для коду, написаному на мові C.

Наступним кроком потрібно створити користувача «ejudge», який використовується для роботи з операційною системою Linux. Дані операції повинні бути виконані на усіх вузлах кластеру (рисунок 3.14).

```
sudo groupadd ejudge
sudo useradd ejudge -s /bin/bash -m -d /home/ejudge -g ejudge
sudo adduser ejudge sudo
```

Рисунок 3.14 – Створення користувача ejudge

Тестуюча система використовує каталоги файлової системи для зберігання вхідних та вихідних параметрів для компілятора, файли

конфігурації турнірів, турнірна таблиця. Дані файли повинні мати властивість високої доступності, тому слід створити дані теки в сховищі Ceph. Даний крок потребує виконання лише на одному вузлі кластеру (рисунок 3.15).

```
sudo mkdir -p /home/judges /home/ceph/judges/test_work
sudo chown ejudge:ejudge /home/ceph/judges /home/ceph/judges/test_work
sudo chmod 0755 /home/judges /home/judges/test_work

sudo mkdir -p /home/ceph/ejudge/cgi-bin
sudo mkdir -p /home/ceph/ejudge/htdocs
sudo chmod 0777 /home/ceph/ejudge/cgi-bin /home/ceph/ejudge/htdocs
```

Рисунок 3.15 – Створення робочих директорій

Для встановлення системи потрібно завантажити останню версію програмного забезпечення з офіційного веб-сайту тестуючої системи – https://ejudge.ru/download/ (рисунок 3.16).

```
$ wget   https://ejudge.ru/download/ejudge-3.8.0.tgz
--2020-05-20 22:39:14--  https://ejudge.ru/download/ejudge-3.8.0.tgz
Resolving ejudge.ru (ejudge.ru)... 89.108.121.5
Connecting to ejudge.ru (ejudge.ru)|89.108.121.5|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7899924 (7.5M) [application/x-gzip]
Saving to: ‘ejudge-3.8.0.tgz’

ejudge-3.8.0.tgz                    100%[=========================================================

2020-05-20 22:39:18 (2.66 MB/s) - ‘ejudge-3.8.0.tgz’ saved [7899924/7899924]

$ tar -xvzf ejudge-3.8.0.tgz
ejudge/
ejudge/cgi-bin/
ejudge/cgi-bin/serve-control.c
ejudge/cgi-bin/users.c
ejudge/cgi-bin/new-client.c
ejudge/xml_utils/
ejudge/xml_utils/err_elem_invalid.c
ejudge/xml_utils/attr_bool.c
ejudge/xml_utils/parse_ip.c
```

Рисунок 3.16 – Завантаження та розпаковка файлів налаштування системи

Система потребує додаткового конфігурування після завантаження та розпаковки.

```
./configure --prefix=/home/ejudge/inst-ejudge \
  --enable-contests-home-dir=/home/judges \
  --with-httpd-cgi-bin-dir=/var/www/ejudge/cgi-bin \
  --with-httpd-htdocs-dir=/var/www/ejudge/htdocs \
  --enable-ajax \
  --enable-charset=utf-8
checking for a BSD-compatible install... /usr/bin/install -c
checking for gawk... gawk
checking for grep that handles long lines and -e... /bin/grep
checking for a sed that does not truncate output... /bin/sed
checking whether ln -s works... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for bison... no
checking for byacc... no
checking for gcc... gcc
checking whether the C compiler works... yes
```

Рисунок 3.17 – Конфігурування системи

Даний скрипт (рисунок 3.17) виконає перевірку необхідних модулів системи та створить файли для подальшого розгортання [21]. Після конфігурування тестуючої системи необхідно виконати встановлення за допомогою команди: «/home/ceph/ejudge/ejudge-setup && /home/ceph/ejudge/ejudge-install.sh». В результаті виконання, виконуючі та допоміжні файли будуть додані до директоріїї /home/ceph/ejudge. Запуск екземпляру ejudge слід виконувати на кожному вузлі кластеру відповідною командою «/home/ceph/ejudge/inst-ejudge/bin/ejudge-control start». Після запуску екземплярів тестуючої системи, система буде доступна за вказаною DNS адресою https://node.kpi-open.kpi.ua/cgi-bin/serve-control (рисунок 3.18).

Рисунок 3.18 – Головна сторінка екземпляру ejudge

3.6 Налаштування CDN для балансування навантаження

CDN (Content Delivery Network) – глобальна мережа, яка служить для прискорення доставки трафіку від постачальників до одержувачів. Обраною CDN є безкоштовний сервіс CloudFlare, який комбінує не лише послуги CDN, а й гарантує захист від DDoS атак, що є безперечним плюсом використання даної системи. Даний сервіс вирішує наступні задачі:

– безпека передачі даних. При роботі через CloudFlare ваш сайт безкоштовно може використовувати SSL-сертифікат, виданий CloudFlare;

– захист сайту від DDoS. CloudFlare самостійно захищає сервіс від атак, не вимагаючи ніяких дій з боку користувача;

– прискорення сайту. Використовуючи CDN, трафік знаходить у найближчому геоположенні до абонентів. Виконується кешування статичних файлів таких як картинки, відео-файли, архіви, Javascript, CSS, що зменшує час завантаження сторінок, зниження навантаження на сервер;

– відмовостійке балансування навантаження на сервіс. Якщо сервіс використовує кілька серверів для обслуговування трафіку, CloudFlare може самостійно розподіляти навантаження [25].

Для підключення серверів до CDN необхідно виконати наступні кроки:

1. Реєстрація в CloudFlare:
   a. Створення облікового запису;
   b. Публікація доменного ім'я сервісу
2. Перенесення DNS записів до NS серверів CloudFlare
3. Створення A записів для вузлів кластеру (node.kpi-open.kpi.ua)
4. Створення головного CNAME запису (kpi-open.kpi.ua) та перенаправлення трафіку на вузли кластеру (node.kpi-open.kpi.ua)

Після виконання даних кроків сервіс стане доступним за головною адресою https://kpi-open.kpi.ua/cgi-bin/serve-control.

ВИСНОВОК ДО РОЗДІЛУ 3

В цьому розділі розглянута процедура налаштування вузла кластеру. Виконана підготовка вузла та розглянутий процес встановлення необхідних для роботи модулів системи на вузлах кластеру. Виконане налаштування мережевої частини кластеру – розглянуте поняття віртуальних мереж VLAN та виділено окремі мережі для кожної частини системи. Розглянуто виконане налаштування мережевого обладнання.

Розглянута процедура встановлення, налаштування та тестування кластеру бази даних Galera, який дозволяє розподілити навантаження до бази даних між вузлами кластеру та підвищити ефективність розроблюваної системи.

Розглянута процедура розгортання система блокового сховища даних Ceph, яка гарантує збереження даних без пошкоджень. Система Ceph дозволила синхронізувати файли на вузлах.

Для кожного вузла кластеру розглянутий процес встановлення та конфігурування екземпляру веб сервера Apache та налаштуванню екземпляру тестуючої системи ejudge. Після конфігурування вузлів кластеру та системи, перенесені DNS записи до сервісу Cloudflare, який надає захист від мережевих атак, та виступає у ролі мережевого балансувальника навантажень між вузлами кластеру.

4 ПРОВЕДЕННЯ ТУРНІРУ З ПРОГРАМУВАННЯ

4.1 Керівництво адміністратора

4.1.1 Створення турніру з програмування

Для створення турніру необхідно перейти до головної сторінки, перейти до сторінки створення турніру та обрати необхідні параметри налаштування.

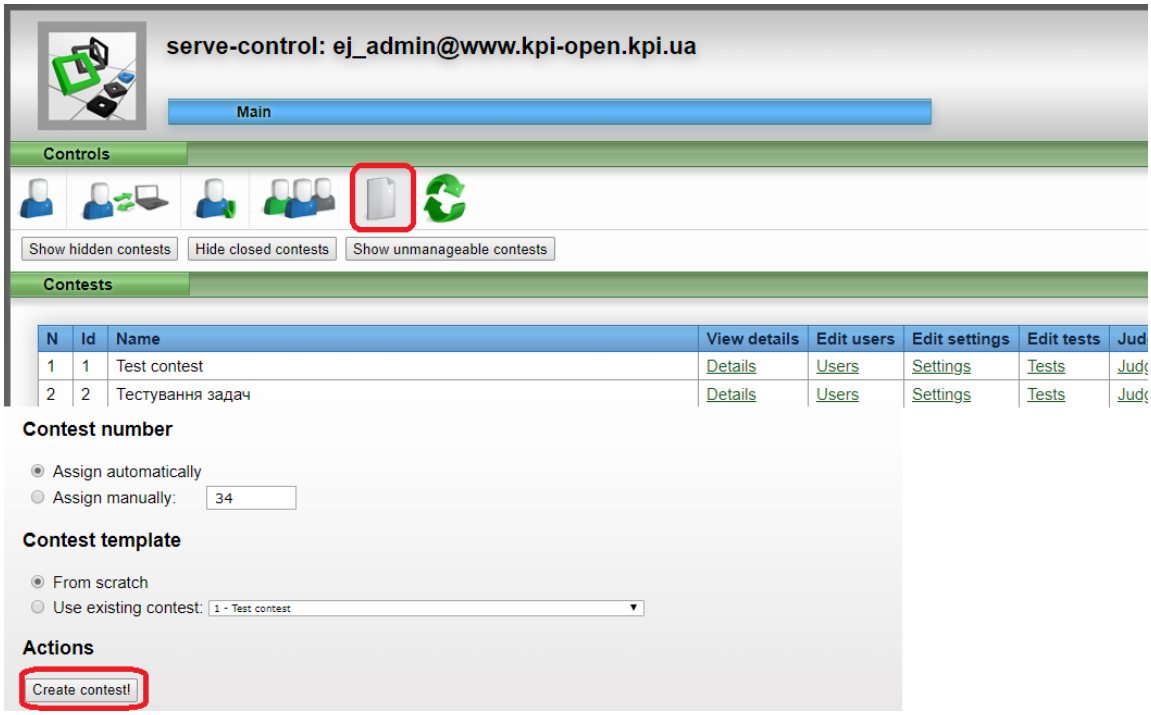


Рисунок 4.1– Створення турніру

Турнір можна створити з шаблону або скопіювати з попередньо проведеної олімпіади (рисунок 4.1). Зазвичай шаблон турніру в залежності від олімпіади відрізняється лише тестовими задачами, інші параметри залишаються без змін. Після створення турніру його слід налаштувати. Конфігурація турніру складається з таких основних модулів:

– глобальних налаштувань турніру

– налаштувань мов програмування

– налаштувань завдань

Модуль глобальних налаштуваннь (рисунок 4.2) відповідає за загальну конфігурацію турніру з програмування, а саме: налаштування турнірної таблиці, визначення посиланнь для учасників турніру, встановлення часових проміжків проведення олімпіади, блокування ір адрес, створення адміністраторів олімпіади.

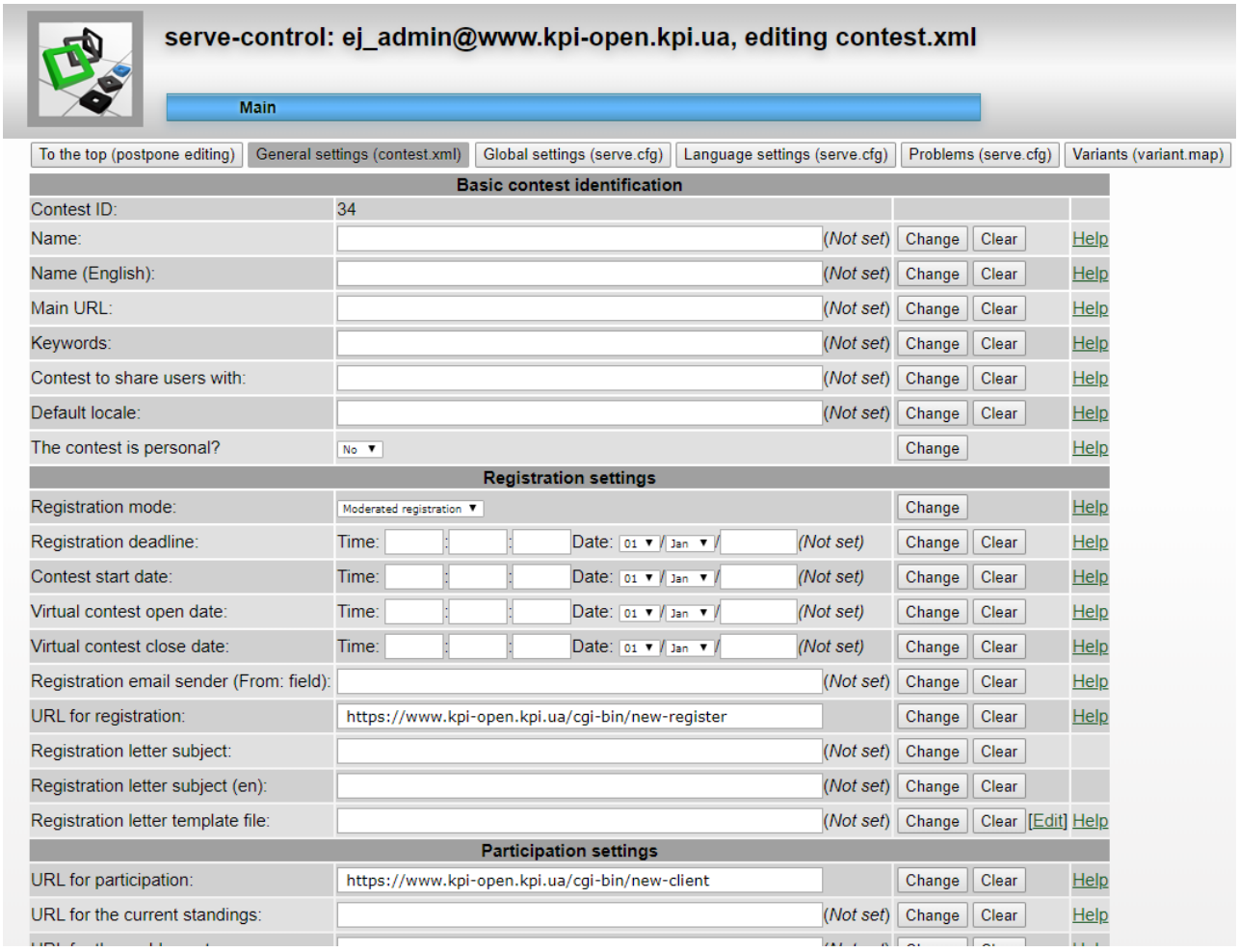


Рисунок 4.2– Модуль глобального конфігурування

Модуль налаштуваннь мов програмування (рисунок 4.3) призначений для вибору доступних мов програмування для використання в турнірі. Дозволяється налаштування кожного компілятора, а саме: кількість виділеної пам'яті, процесорного часу, встановлення захисту від втручання у систему, додавання допоміжних бібліотек, обмеження на розмір програмного коду.

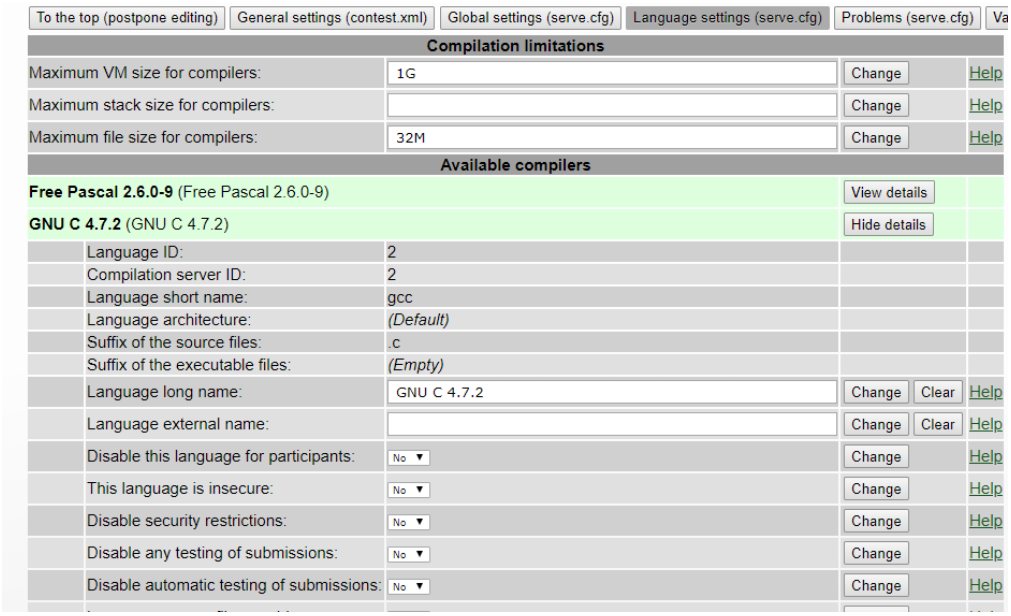


Рисунок 4.3– Модуль налаштування мов програмування

Модуль налаштування завдань (рисунок 4.4) відповідає за безпосереднє налаштування задач, а саме: вибір маски вхідних файлів перевірки, вибір маски вихідних файлів перевірки, вибір методу перевірки завдань, встановлення лімітів на використання пам’яті та процесорного часу. Цей модуль дозволяє додавати на сервер скомпільовані файли перевірки, які будуть перевіряти вихідні значення коду наданого учасниками олімпіади.

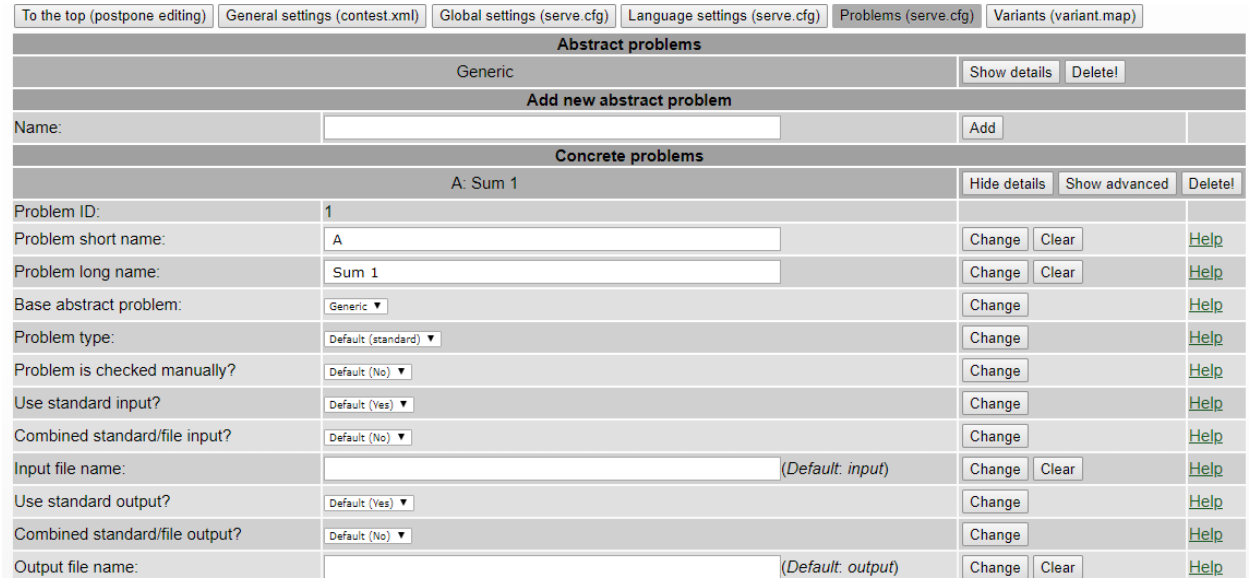


Рисунок 4.4– Модуль налаштування мов програмування

Коли конфігурування модулів завершене, необхідно натиснути клавішу «COMMIT changes!», яка виконає перевірку значень у конфігурації та виконає збереження конфігурації до серверу (рисунок 4.5).

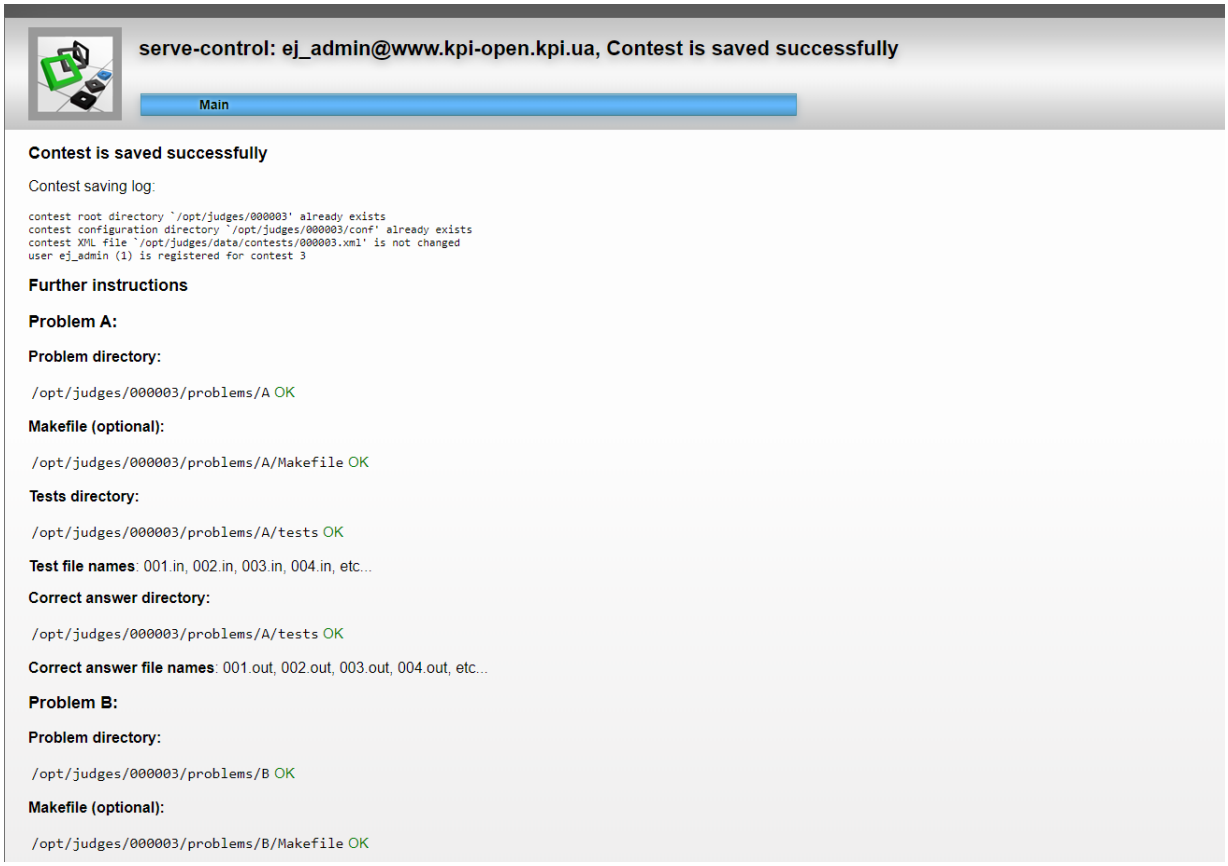


Рисунок 4.5– Збереження налаштувань

4.1.2 Генерування таблиці результатів

Тестуюча система ejudge підтримує створення та оновлення таблиці результатів у режимі реального часу. Для створення турнірної таблиці необхідно перейти до налаштування турніру та обрати конфігурацію «General settings (contest.xml)» далі необхідно перейти до налаштувань участі, у полі «URL for the current standings» вибрати URL адресу за якою буде доступна таблиця результатів та зберегти зміни натиснувши клавішу «Change» та «COMMIT changes!» (рисунок 4.6).

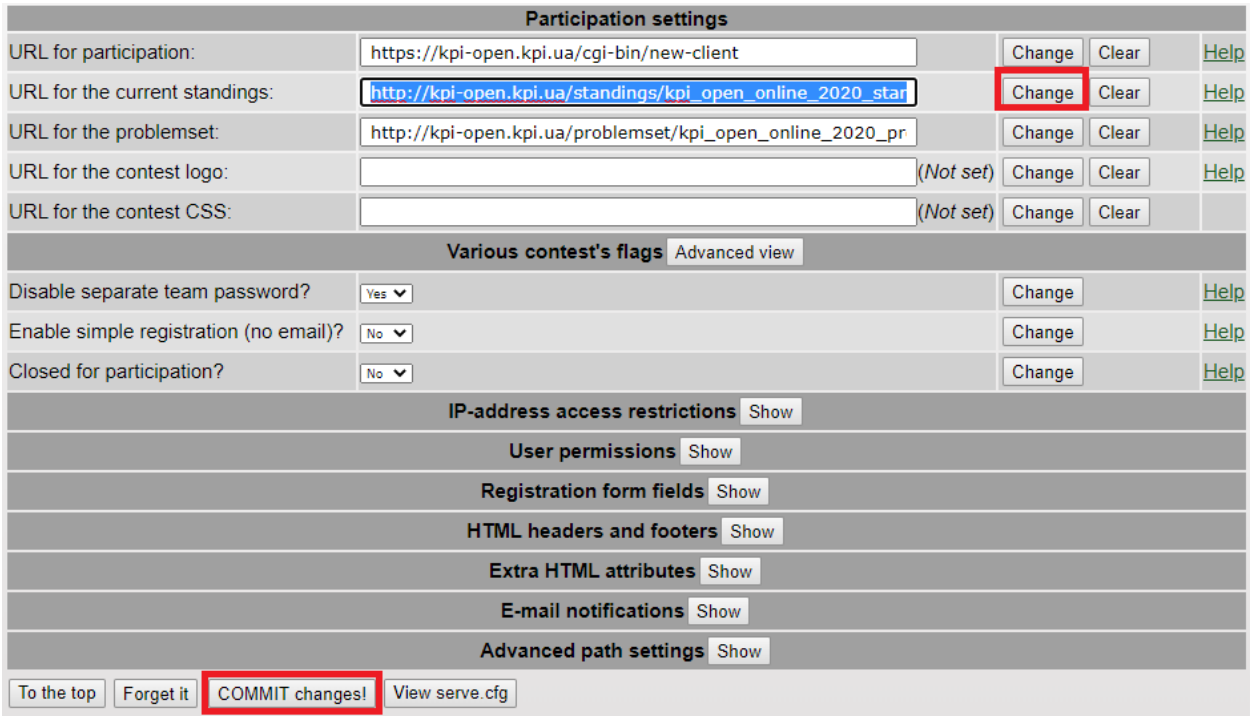
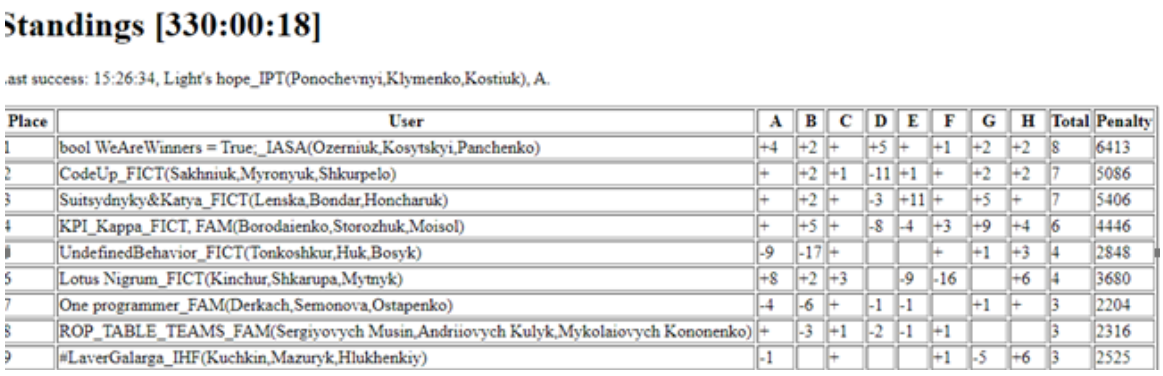


Рисунок 4.6 – Створення турнірної таблиці

Після збереження налаштувань, турнірна таблиця буде доступною за вказаною адресою, після початку турніру система створить сторінку, на яку будуть додаватися результати учасників олімпіади (рисунок 4.7).



Standings [330:00:18]

ast success: 15:26:34, Light's hope_IPT(Ponochevnyi,Klymenko,Kostiuk), A.

Place	User	A	B	C	D	E	F	G	H	Total	Penalty
1	bool WeAreWinners = True;_IASA(Ozerniuk,Kosytskyi,Panchenko)	+4	+2	+	+5	+	+1	+2	+2	8	6413
2	CodeUp_FICT(Sakhniuk,Myronyuk,Shkurpelo)	+	+2	+1	-11	+1	+	+2	+2	7	5086
3	Suitsydnyky&Katya_FICT(Lenska,Bondar,Honcharuk)	+	+2	+	-3	+11	+	+5	+	7	5406
4	KPI_Kappa_FICT, FAM(Borodaienko,Storozhuk,Moisol)	+	+5	+	-8	-4	+3	+9	+4	6	4446
8	UndefinedBehavior_FICT(Tonkoshkur,Huk,Bosyk)	-9	-17	+			+	+1	+3	4	2848
5	Lotus Nigrum_FICT(Kinchur,Shkarupa,Mytnyk)	+8	+2	+3		-9	-16		+6	4	3680
7	One programmer_FAM(Derkach,Semonova,Ostapenko)	-4	-6	+	-1	-1		+1	+	3	2204
8	ROP_TABLE_TEAMS_FAM(Sergiyovych Musin,Andriiovych Kulyk,Mykolaiovych Kononenko)	+	-3	+1	-2	-1	+1			3	2316
9	#LaverGalarga_IHF(Kuchkin,Mazuryk,Hlukhenkiy)	-1		+			+1	-5	+6	3	2525

Рисунок 4.7– Вигляд турнірної таблиці під час проведення олімпіади

На турнірній таблиці відображаються та оновлюються наступні поля: назва команди, статус виконання завдань, місце по виконаним задачам серед інших учасників.

4.1.3 Завантаження завдань

Для створення завдання необхідно перейти до налаштування турніру та обрати конфігурацію «Problems (serve.cfg)», перейти до модулю «Concrete problems» та натиснути на клавішу «Add» (рисунок 4.8).

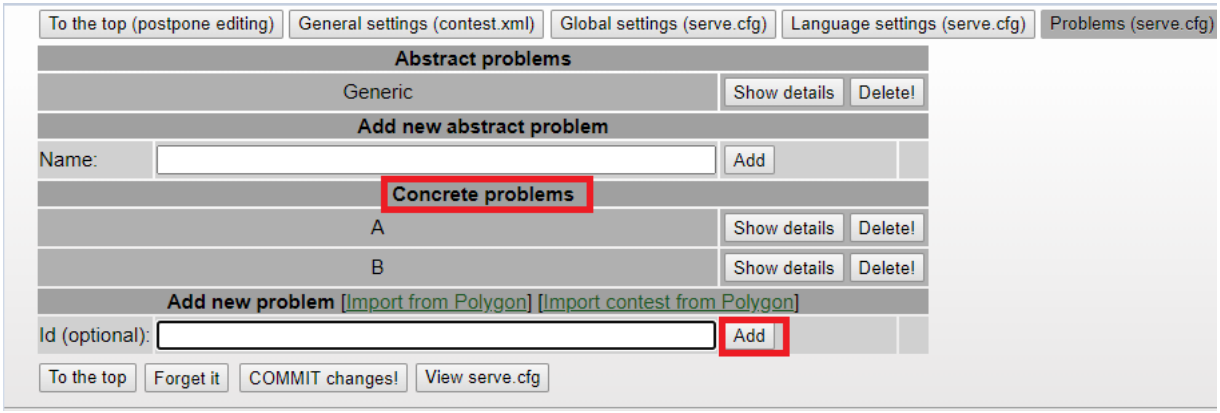


Рисунок 4.8– Створення завдання

Для завдання необхідно налаштувати обмеження по використанні оперативної пам’яті (Maximum virtual memory size) та максимальному процесорному часу (Processor time limit), вибрати метод перевірки надісланого коду (Standart checker), зазвичай вибирається порівняння двох чисел: порівнюються числа з завантажених файлів перевірки та стандартного виводу користувацької програми.

Для даних файлів необхідно вказати маску – за маскою тестуюча система отримає файли перевірки з директорії сховища (рисунок 4.9).

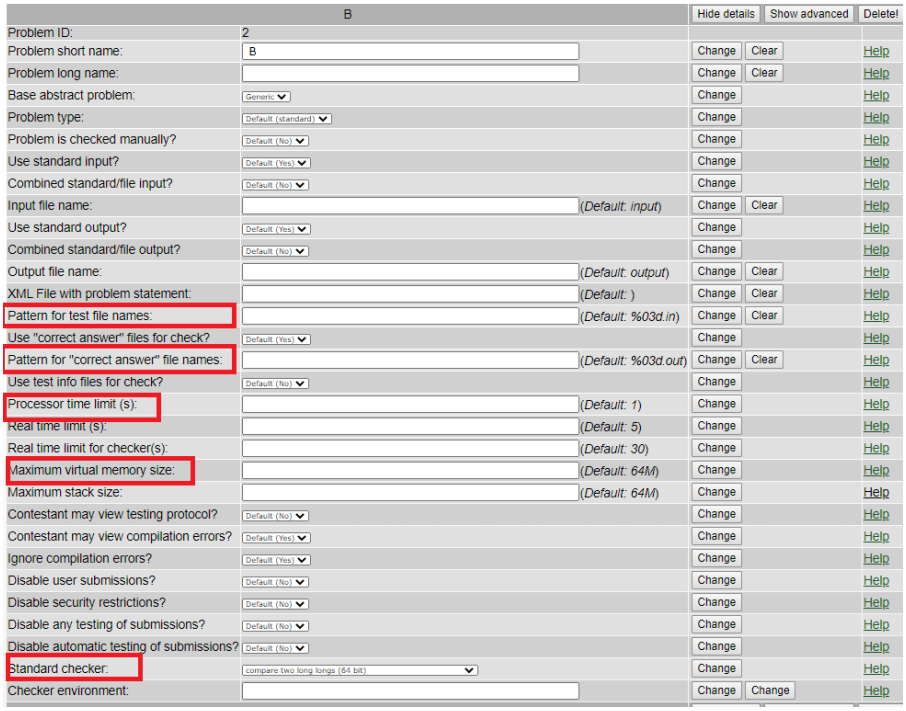


Рисунок 4.9– Налаштування параметрів завдання

Для роботи налаштованого завдання необхідно завантажити файли перевірки. Для цього слід перейти до налаштуваннь задач обраного турніру, обрати завдання та додати новий тест для завдання (рисунок 4.10).

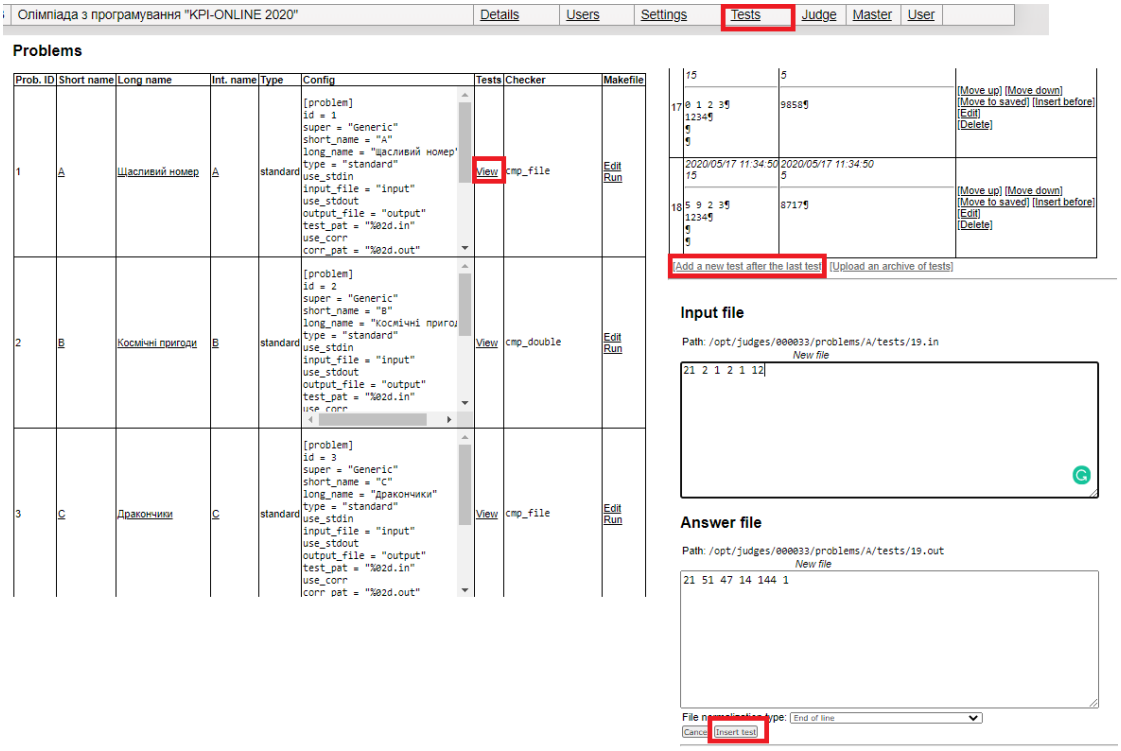


Рисунок 4.10– Завантаження файлів перевірки завдання

Файл автоматично включається в список тестів, які будуть виконуватися під час перевірки надісланого коду учасника після додавання файлу перевірки до системи. Після створення та завантаження завдання необхідно перевірити файлу перевірки за допомогою еталонної програми, створенної автором завдання. Для цього необхідно відправити завдання на перевірку, як учасник турніру.

4.1.4 Робота з користувачами

Система ejudge дозволяє створювати облікові записи учасників олімпіади або надавати інтерактивне посилання до самостійної реєстрації. Для створення користувача необхідно перейти за посиланням «Users» та вибрати функцію «User creation operations» (рисунок 4.11) Створення облікових записів учасників олімпіади). Дана функція дозволяє створити декілька користувачів одночасно, що є безперечним плюсом для адміністратора.

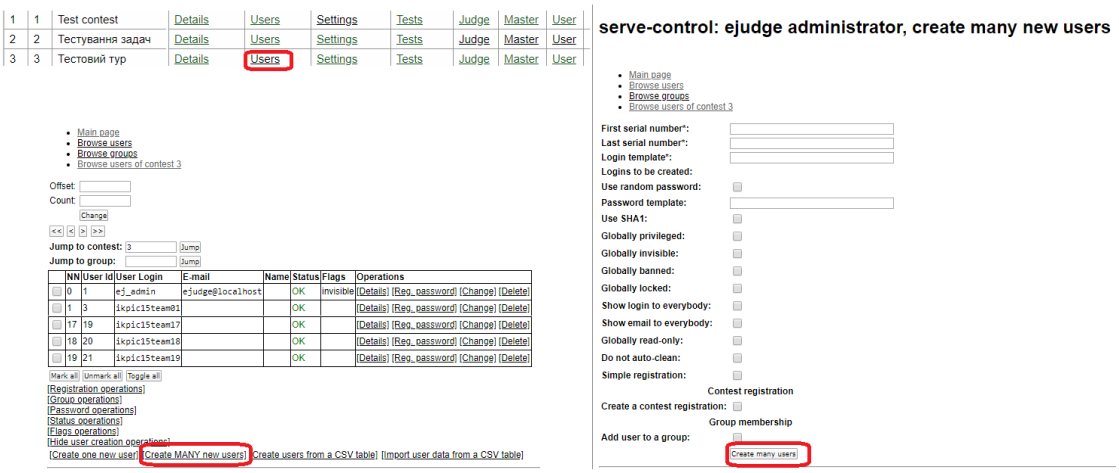


Рисунок 4.11– Створення облікових записів учасників олімпіади

Кожен користувач системи має різні можливості, які представлені за допомогою діаграми претендентів ІА62.010БАК.005 Д4. У тестуючій системі можна виділити 4 основні ролі: тренер, учасник, організатор, адміністратор.

Роль організатора має ряд таких можливостей:

– перегляд турнірної таблиці;

– створення користувачів;

– повторна перевірка рішень;

– запуск турніру.

Роль адміністратора має такі можливості:

– перегляд турнірної таблиці;

– створення користувачів;

– повторна перевірка рішень;

– запуск турніру;

– конфігурування турніру;

– зупинка турніру.

Роль тренера має можливості тільки перегляду турнірної таблиці.

Роль учасника має такі можливості:

– відправка задач на перевірку;

– перегляд турнірної таблиці.

Даний розподіл обмежує права користувачів, тим самим підвищуючи рівень безпеку робочої системи. Розмежування ролей дозволяє зробити процес конфігурування та проведення олімпіади зручнішим завдяки представленню тільки необхідних можливостей кожному типу користувачів системи.

4.1.5 Запуск олімпіади

Для запуску турніру з програмування слід перейти до сторінки адміністратору турніру, встановити час проведення олімпіади та натиснути на клавішу «Start» (рисунок 4.12). Коли час проведення олімпіади закінчиться система автоматично зупинить турнір.

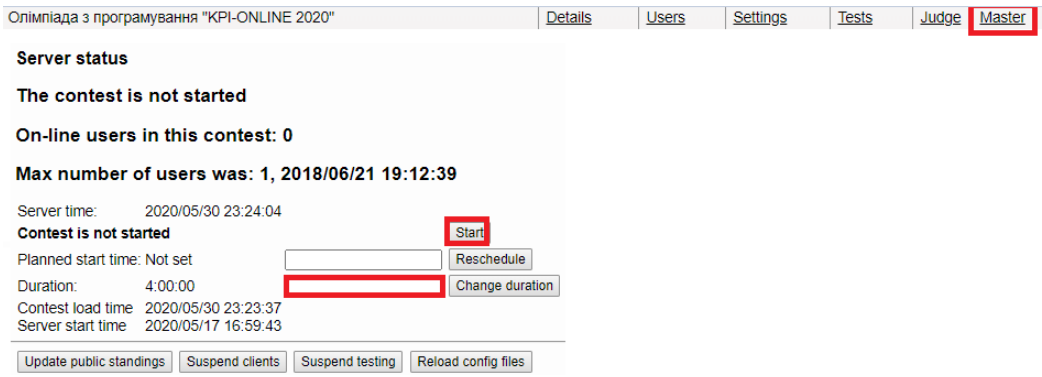


Рисунок 4.12– Запуск турніру з програмування

Після запуску турніру, адміністратор може бачити результат проходження тестів та може отримати детальну інформацію щодо пройдених тестів. Для того, щоб отримати інформацію щодо пройдених тестів та отримати копію надісланого програмного коду необхідно перейти до сторінки адміністратору турніру, вибрати спробу перевірки програми та натиснути на посилання «View»

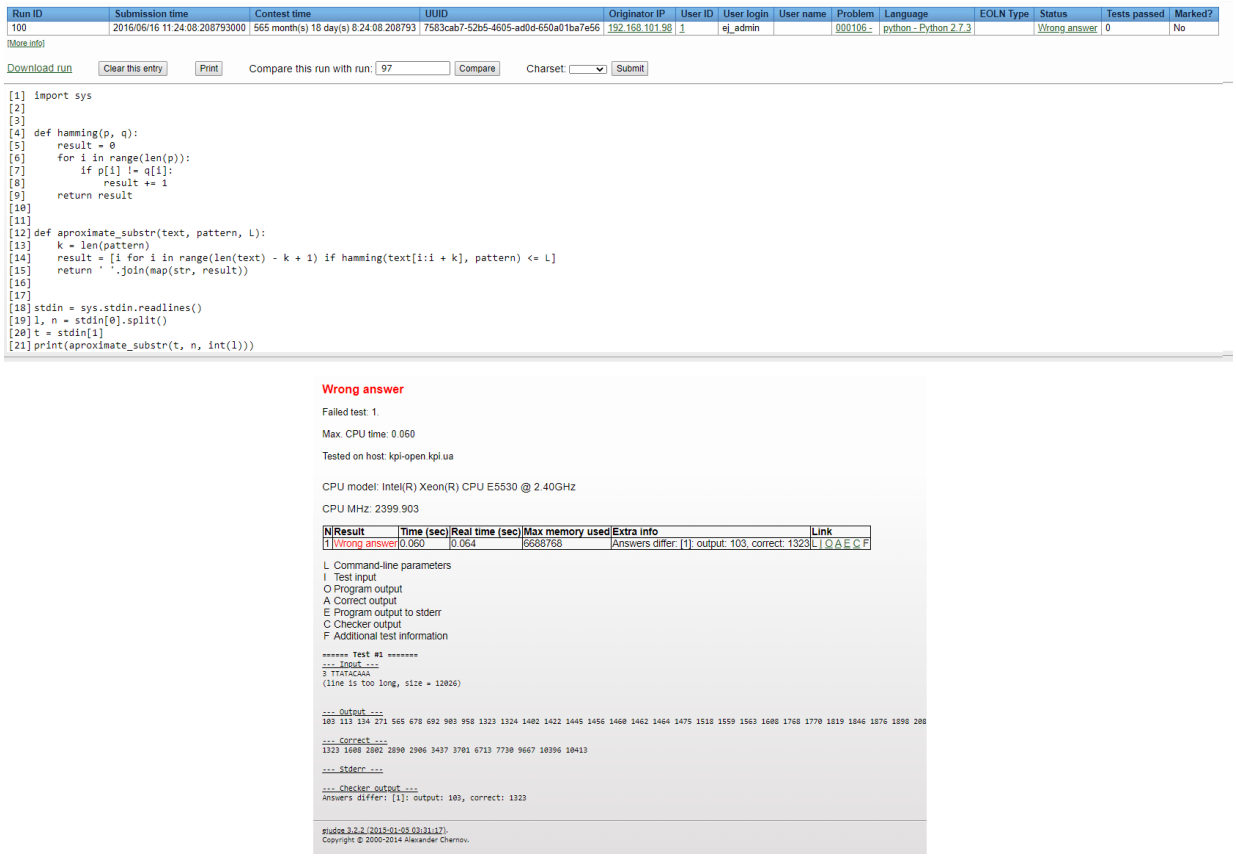


Рисунок 4.13– Отримання детальної інформації щодо надісланого розв’язку

На даних сторінках (рисунок 4.13) можна отримати таку системну інформацію: IP адреса, з якої надсилалося завдання, ідентифікатор користувача, мова програмування, статус обробки, час відправки рішення, ідентифікатор пройдених тестів перевірки.

4.2 Керівництво учасника

4.2.1 Зміна паролю

Після успішної авторизації учасник повинен змінити пароль, для уникнення проблем з несанкціонованим доступом. Для цього необхідно перейти до сторінки налаштувань та змінити пароль за замовчуванням, після внесених змін слід натиснути на клавішу «Change» (рисунок 4.14).

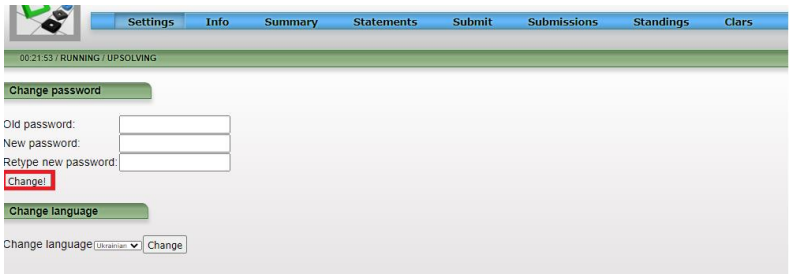


Рисунок 4.14 – Зміна паролю учасника

4.2.2 Відправка завдань до тестуючої системи

Для відправки завдань необхідно перейти до розділу «Submit», обрати задачу та мову програмування та натиснути на клавішу «Submit» (рисунок 4.15).

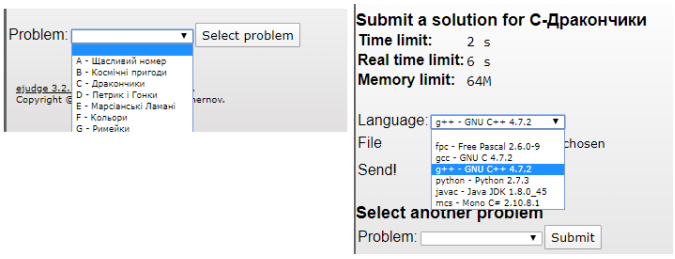
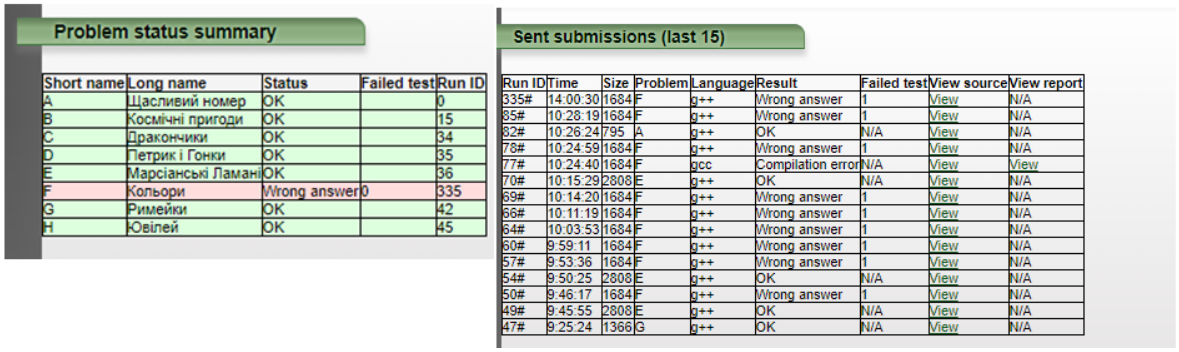


Рисунок 4.15 – Надсилання коду на перевірку

Після цього програмний код учасника буде надісланий до тестуючої системи на компіляцію, виконання та перевірку. Учасник має змогу побачити статус виконаних завдань у розділі «Summary» та «Submissions» (рисунок 4.16).



Problem status summary

Short name	Long name	Status	Failed test	Run ID
A	Щасливий номер	OK		0
B	Космічні пригоди	OK		15
C	Дракончики	OK		34
D	Петрик і Гонки	OK		35
E	Марсіанські Ламані	OK		36
F	Кольори	Wrong answer	0	335
G	Римейки	OK		42
H	Ювілей	OK		45

Sent submissions (last 15)

Run ID	Time	Size	Problem	Language	Result	Failed test	View source	View report
335#	14:00:30	1684	F	g++	Wrong answer	1	View	N/A
85#	10:28:19	1684	F	g++	Wrong answer	1	View	N/A
82#	10:26:24	795	A	g++	OK	N/A	View	N/A
78#	10:24:59	1684	F	g++	Wrong answer	1	View	N/A
77#	10:24:40	1684	F	gcc	Compilation error	N/A	View	View
70#	10:15:29	2808	E	g++	OK	N/A	View	N/A
69#	10:14:20	1684	F	g++	Wrong answer	1	View	N/A
66#	10:11:19	1684	F	g++	Wrong answer	1	View	N/A
64#	10:03:53	1684	F	g++	Wrong answer	1	View	N/A
60#	9:59:11	1684	F	g++	Wrong answer	1	View	N/A
57#	9:53:36	1684	F	g++	Wrong answer	1	View	N/A
54#	9:50:25	2808	E	g++	OK	N/A	View	N/A
50#	9:46:17	1684	F	g++	Wrong answer	1	View	N/A
49#	9:45:55	2808	E	g++	OK	N/A	View	N/A
47#	9:25:24	1366	G	g++	OK	N/A	View	N/A

Рисунок 4.16 – Результати перевірки коду

Вирішені задачі будуть підсвічені тестуючою системою зеленим кольором, задачі, які не пройшли перевірку – червоним (рисунок 4.16). Кількість використаних спроб впливає на кінцевий результат в таблиці результатів. Учасник з меншою кількісттю спроб вирішення завдань буде мати перевагу над іншими.

4.3 Процес проведення міжнародної олімпіади

Основою для проведення олімпіади є тестуюча система. Тестуюча система виконує функції перевірки завдань, формування туртірної таблиці, визначення мов програмування, формування обмежень для програм створених учасниками.

Проведення міжнародної олімпіади складається з декількох етапів починається з етапу підготовки турніру тестуючої системи. До цього етапу входить:

– розробка завдань ;

– розробка перевіряючих програм та файлів;

– створення турніру у тестуючій системі;

– створення завдань у тестуючій системі;

– створення облікових записів учасників у тестуючій системі;

– підготовка робочого місця учасника;

– завантаження завдань до тестуючої системи;

– завантаження файлів перевірки завдань;

– генерування таблиці результатів;

– перевірка роботи системи за допомогою відправки та подальшої перевірки завданнь за допомогою еталонного коду автору в тестуючій системі.

Наступний етап потребує перевірку системи на відмовостійкість. На даному етапі необхідно запустити турнір та виконати агресивне тестування: виконати відмову вузла, відмову операційної системи, втрату енергоживлення та інші. Головна ідея даного тестування – розробити план дій щодо усунення проблем у майбутньому та отримати реальні дані про ступінь готовності системи до проведення міжнародної олімпіади. Якщо робота системи не відновилася після виконання тестування – це вказує на те, що система потребує негайної модернізації або створенні інструкцій для адміністратора системи по відновленню.

Виконання даних кроків може гарантувати працездатність та готовність системи до переходу до третього етапу. Даний етап проведення олімпіади є її безпосереднього проведення та складається з наступних кроків:

– розміщення учасників олімпіади;

– видача авторизаційних даних учасникам;

– запуск турніру в тестуючій системі;

– завершення турніру в тестуючій системі.

На даному етапі можуть виникнути проблеми, які не були протестовані на попередніх етапах. Ці проблеми майже неможливо спрогнозувати, їх рішення потребує швидкого реагування організаційного комітету та

адміністраторів системи. Дані проблеми можуть бути пов’язані з тестуючою системою або апаратним забезпеченням тестуючої системи. Зазвичай вони вирішуються перезавантаженням проблемного вузла та можуть створити затримку системи у 1-7 хв, що не є критичним.

4.4 Тестування розробленої системи

Для тестування системи необхідно виконати завдання першого етапу проведення олімпіади, а саме підготувати середовище для участі у турнірі з програмування [26]. Тестування в цілому складається з двох кроків:

– тестування системи при ідеальних умовах;

– тестування системи на межі відмови.

Перший крок тестування роботи системи – це запуск турніру та завантаження програмного коду до системи та подальша перевірка надісланого рішення. При виконанні даного кроку слід перевірити занесення відповідних даних до бази даних та розподіленої файлової системи. Даний крок дозволить перевірити працездатність системи в цілому та відобразить стан роботи розподіленої файлової системи, бази даних та самої тестуючої системи.

Другий крок потребує створення ситуацій, коли система має схильність до відмов. Для перевірки відмовостійкості слід змоделювати катастрофічні відмови апаратної та програмної складової кластеру.

При виході з ладу блоку живлення, жорсткого диску, операційної системи – вузол кластеру стає недоступним для інших вузлів, що може призвести до відмови системи. Після вимкнення вузла система повинна продовжити функціонування. Для кластеру відмова вузла спричиняє відмову екземплярів бази даних, блокової системи, тестуючої системи на вузлі кластеру.

У результаті моделювання відмови вузлу кластеру система продовжила обробляти запити користувачів та виконувати заданий функціонал. Після

підключення вузла до кластеру, виконана автоматична реплікація бази даних та блокового сховища. Після виконаних дій вузол повністю відновив свою роботу не пошкодивши дані та не вплинувши на кінцеву систему.

При виході з ладу блоку живлення або апаратному збою комутатора, комутатор перестає обробляти мережеві дані, що призводить до втрати комунікації між вузлами кластеру. Для уникнення даної ситуації впроваджено резервування каналів передачі даних. При моделюванні виходу з ладу комутатора, інший продовжував виконувати задані функції, та за допомогою сконфігурованого агрегування каналів в системі вузла, відмови системи не спостерігалось.

При відмові централізованої системи живлення сервери, які до неї приєднані можуть бути вимкнені та стати недоступними для кластеру. При моделюванні даної проблеми вузли кластеру продовжили свою роботу завдяки резервній системі живлення та використання безперебійних джерел живлення.

При відмові блоку живлення чи безперебійного джерела вузол може припинити свою роботу. При моделюванні даної проблеми було відключено один блок живлення сервер. Завдяки резервуванню елементів живлення сервер продовжив свою роботи без переривань завдяки резервному блоку живлення.

Проведене тестування показало високу відмовостійкість розробленої системи в умовах відмови апаратної та програмної складової.

ВИСНОВОК ДО РОЗДІЛУ 4

В даному розділі розроблено керівництво користувача та системного адміністратора. Розроблена методика для проведення міжнародної олімпіади, а саме:

– підготовка до проведення олімпіади;

– перевірка роботи системи;

– безпосереднє проведення олімпіади.

Виконане тестування роботи системи та проведене моделювання ситуацій, коли тестуюча система схильна до відмов. При проведенні тестування, розроблювана система зберігала працездатність на весь період виконуваних робіт. При спроектованих ситуація тестуюча система продовжила свою роботу без затримки.

Агресивне тестування підтверджує готовність системи до проведення олімпіад міжнародного рівня. Адже головною вимогою до тестуючої системи при проведенні олімпіад міжнародного рівня є схильність до відновлення після відмови та стійкість до впливу зовнішніх факторів.

ВИСНОВКИ

В ході виконання дипломного проєкту досліджені тестуючі системи. На основі порівняння їх характеристик та особливостей створені вимоги до розроблюваної системи.

В результаті огляду існуючих тестуючих систем знайдено спільну проблему тестуючих систем– відсутність відмовостійкості. Для модифікації обрана система ejudge, яка є гнучкою до модифікацій, має відкритий вихідний код, представляє гнучкий до налаштування функціонал.

Отримані необхідні знання про підходи до організації високої відмовостійкості. Визначені вимоги до проектування додатків. Для побудови кластеру використана топологія гібридної реплікації бази даних Galera та система блокового сховища Ceph.

Створені графічні матеріали дозволили побудувати та сконфігурувати розроблювану систему. Розроблена та розгорнута відмовостійка архітектура для тестуючої системи згідно визначених умов.

Розглянутий процес проведення міжнародних олімпіад, що дозволило окреслити план тестування для розробленої системи.

Тестуюча система справно функціонує і вирішує поставлені перед нею завдання.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ролик А.И. Управление корпоративной ИТ-инфраструктурой / А.И. Ролик, С.Ф. Теленик, М.В. Ясочка // К.: Наукова думка, 2018. – 576 с. . – Режим доступу до ресурсу: https://acts.kpi.ua/app/uploads/2020/05/rolik_teleni_yasochka_uiti.pdf

2. Денисов Д. В. АРХИТЕКТУРА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ТЕСТИРОВАНИЯ РЕШЕНИЙ ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ [Електронний ресурс] / Денис Валерійович Денисов // Компьютерные инструменты в образовании / Денис Валерійович Денисов., 2014. – С. 50–58. – Режим доступу до ресурсу: https://cyberleninka.ru/article/n/arhitektura-avtomatizirovannoy-sistemy-testirovaniya-resheniy-zadach-po-programmirovaniyu/viewer

3. Выбор системы автоматизированного тестирования решений задач по программированию[Електронний ресурс] // International Journal of Open Information Technologies / 2016. – С. 38–43. - Режим доступу до ресурсу: http://injoit.org/index.php/j1/article/viewFile/300/249

4. Самощенко, Ю. Ю. Исследование эффективности автоматизированной проверки решений при проведении олимпиад по программированию / Ю. Ю. Самощенко. – Текст : безпосередній // Молодой ученый. – 2016. – № 11 (115). – С. 223-226. - Режим доступу до ресурсу: https://moluch.ru/archive/115/31127/

5. Настройка системы Contester для школьной олимпиады. Часть 1 [Електронний ресурс]. – Режим доступу до ресурсу: https://19dx.ru/2016/02/nastrojka-sistemy-contester-dlya-shkolnoj-olimpiady-chast-1/

6. Документація системи PCMS2 [Електронний ресурс]. – Режим доступу до ресурсу: http://neerc.ifmo.ru/trains/information/index.html

7. Система проведения соревнований ejudge [Електронний ресурс]. – 2007. – Режим доступу до ресурсу: https://ejudge.pit.org.ua/files/ru.pdf

8. PC² [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/PC%C2%B2.

9. Contest Administrator's Installation and Configuration Guide [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://pc2.ecs.csus.edu/doc/v9/9.6.0/pc2v9AdminGuide.pdf.

10. Computer cluster [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Computer_cluster.

11. High-availability cluster [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/High-availability_cluster

12. Кластеризация [Електронний ресурс]. – 2019. https://stefaniuk.website/all/clusters/

13. Ceph vs GlusterFS vs MooseFS vs HDFS vs DRBD [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://computingforgeeks.com/ceph-vs-glusterfs-vs-moosefs-vs-hdfs-vs-drbd/.

14. Знакомство с хранилищем Ceph в картинках [Електронний ресурс]. – 2016.– https://habr.com/ru/post/313644/

15. Документація сховища HDFS [Електронний ресурс]. – https://www.tutorialspoint.com/hadoop/

16. High availability with the Distributed Replicated Block Device [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: https://developer.ibm.com/technologies/linux/tutorials/l-drbd/.

17. An Overview of MySQL Database High Availability [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://severalnines.com/blog/overview-mysql-database-high-availability.

18. MySQL Replication for High Availability [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://severalnines.com/resources/database-management-tutorials/mysql-replication-high-availability-tutorial.

19. Galera Cluster [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://ru.bmstu.wiki/Galera_Cluster.

20. Аппаратное обеспечение кластера высокой готовности [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: http://lab50.net/%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5-%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D0%BA%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80%D0%B0-%D0%B2%D1%8B%D1%81/.

21. Документація тестуючої системи ejudge [Електронний ресурс]. – Режим доступу до ресурсу: https://ejudge.ru/wiki/index.php/Main_Page

22. Установка MariaDB Galera для управления кластером репликации [Електронний ресурс] – Режим доступу до ресурсу: https://system-admins.ru/ustanovka-mariadb-galera-dlya-upravleniya-klasterom-replikacii/

23. Базовая настройка кластера CephFS на простом примере [Електронний ресурс] – Режим доступу до ресурсу: https://specialistoff.net/page/325.

24. HTTPS [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTTPS.

25. Защита, оптимизация и повышение производительности сайта с помощью CDN CloudFlare [Електронний ресурс] – Режим доступу до ресурсу: https://netpoint-dc.com/blog/zashita-optimizaciya-povishenie-proizvoditelnosti-saita-s-pomoshiyu-cdn-cloudflare/

26. Липаєв, В. В. Надежность и функциональная безопасность комплексов программ реального времени монографія/ В. В. Липаєв. – М.-Берлін Діректмеліа, 2015. – 281с